

**TRƯỜNG ĐẠI HỌC THỦY LỢI**  
**KHOA CÔNG NGHỆ THÔNG TIN**



# **GIÁO TRÌNH**

## **THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG**

Hà Nội, 2.2025

# MỤC LỤC

CHƯƠNG 1. Làm quen.....	4
Bài 1) Tạo ứng dụng đầu tiên.....	4
1.1) Android Studio và Hello World.....	4
1.2) Giao diện người dùng tương tác đầu tiên.....	5
1.3) Trình chỉnh sửa bố cục.....	5
1.4) Văn bản và các chế độ cuộn.....	5
1.5) Tài nguyên có sẵn.....	5
Bài 2) Activities.....	5
2.1) Activity và Intent.....	5
2.2) Vòng đời của Activity và trạng thái.....	5
2.3) Intent ngầm định.....	5
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ.....	5
3.1) Trình gỡ lỗi.....	5
3.2) Kiểm thử đơn vị.....	5
3.3) Thư viện hỗ trợ.....	5
CHƯƠNG 2. Trải nghiệm người dùng.....	6
Bài 1) Tương tác người dùng.....	6
1.1) Hình ảnh có thể chọn.....	6
1.2) Các điều khiển nhập liệu.....	6
1.3) Menu và bộ chọn.....	6
1.4) Điều hướng người dùng.....	6
1.5) RecyclerView.....	6
Bài 2) Trải nghiệm người dùng thú vị.....	6
2.1) Hình vẽ, định kiểu và chủ đề.....	6
2.2) Thẻ và màu sắc.....	6
2.3) Bố cục thích ứng.....	6
Bài 3) Kiểm thử giao diện người dùng.....	6

3.1) Espresso cho việc kiểm tra UI.....	6
CHƯƠNG 3. Làm việc trong nền.....	6
Bài 1) Các tác vụ nền.....	6
1.1) AsyncTask.....	6
1.2) AsyncTask và AsyncTaskLoader.....	6
1.3) Broadcast receivers.....	6
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	6
2.1) Thông báo.....	6
2.2) Trình quản lý cảnh báo.....	6
2.3) JobScheduler.....	6
CHƯƠNG 4. Lưu dữ liệu người dùng.....	7
Bài 1) Tùy chọn và cài đặt.....	7
1.1) Shared preferences.....	7
1.2) Cài đặt ứng dụng.....	7
Bài 2) Lưu trữ dữ liệu với Room.....	7
2.1) Room, LiveData và ViewModel.....	7
2.2) Room, LiveData và ViewModel.....	7

## CHƯƠNG 1. LÀM QUEN

### Bài 1) Tạo ứng dụng đầu tiên

#### 1.1) Android Studio và Hello World

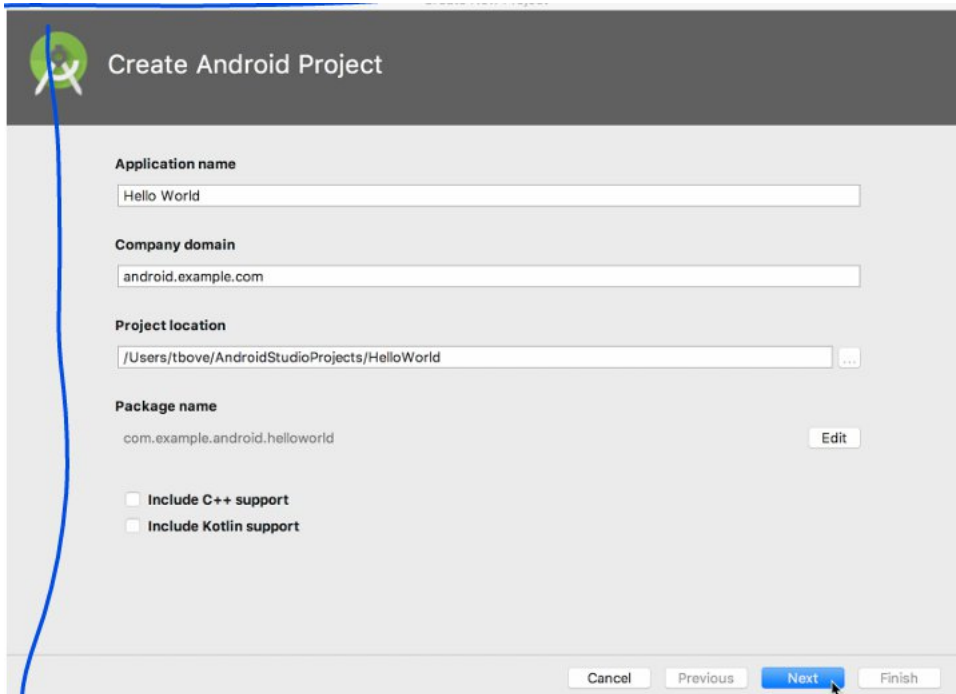
### Giới thiệu

Trong bài thực hành này, bạn sẽ tìm hiểu cách cài đặt Android Studio, môi trường phát triển Android. Bạn cũng sẽ tạo và chạy ứng dụng Android đầu tiên của mình, Hello World, trên một trình giả lập và trên một thiết bị vật lý.

### Những gì Bạn nên biết

Bạn nên có khả năng:

- Hiểu quy trình phát triển phần mềm tổng quát cho các ứng dụng lập trình hướng đối tượng sử dụng một IDE (môi trường phát triển tích hợp) như Android Studio.
- Chứng minh rằng bạn có ít nhất 1-3 năm kinh nghiệm trong lập trình hướng đối tượng, với một phần trong số đó tập trung vào ngôn ngữ lập trình Java. (Các bài thực hành này sẽ không giải thích về lập trình hướng đối tượng hoặc ngôn ngữ Java.



### Những gì Bạn sẽ cần:

- Một máy tính chạy Windows hoặc Linux, hoặc một Mac chạy macOS. Xem trang tải xuống Android Studio để biết yêu cầu hệ thống cập nhật.
- Truy cập Internet hoặc một phương pháp thay thế để tải các cài đặt mới nhất của Android Studio và Java lên máy tính của bạn.

### Những gì bạn sẽ học

- Cách cài đặt và sử dụng IDE Android Studio.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo một dự án Android từ một mẫu.
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

### Những gì bạn sẽ làm

- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.
- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.
- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

## Tổng quan về ứng dụng

Sau khi bạn cài thành công Android Studio , bạn sẽ tạo một dự án mới từ một mẫu cho ứng dụng Hello World . Ứng dụng đơn giản này hiển thị chuỗi “Hello World” trên màn hình của máy ảo Android hoặc thiết bị thật.

Ứng dụng khi hoàn thành sẽ trông như thế này:



## Bước 1: Cài Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp hoàn chỉnh (IDE), bao gồm một trình chỉnh sửa mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ hỗ trợ phát triển, gỡ lỗi, kiểm thử và tối ưu hiệu suất, giúp quá trình phát triển ứng dụng trở nên nhanh chóng và dễ dàng hơn.

Bạn có thể kiểm thử ứng dụng của mình trên nhiều trình giả lập được cấu hình sẵn hoặc trên thiết bị di động của bạn, đồng thời có thể xây dựng ứng dụng hoàn chỉnh và phát hành trên **Google Play Store**.

**Lưu ý:** Android Studio liên tục được cải tiến. Để biết thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, hãy truy cập **Android Studio**.

Android Studio khả dụng cho máy tính chạy Windows or Linux , và Macs chạy MacOS. OpenJDK(Java Development Kit) mới nhất được tích hợp sẵn trong Android Studio.

Để bắt đầu với Android Studio , trước tiên kiểm tra yêu cầu hệ thống để đảm bảo thiết bị của bạn đáp ứng đủ điều kiện . Trình cài đặt tương tự với tất cả nền tảng . Chỉ khác nhau vài điều được ghi dưới đây :

1. Truy cập trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận cấu hình mặc định ở tất cả các bước và đảm bảo rằng tất cả các thành phần cần thiết đều được chọn để cài đặt.
3. Sau khi quá trình cài đặt hoàn tất, Setup Wizard sẽ tự động tải xuống và cài đặt thêm một số thành phần bổ sung, bao gồm Android SDK. Hãy kiên nhẫn, vì quá trình này có thể mất một chút thời gian tùy vào tốc độ Internet của bạn, và một số bước có thể lặp lại.
4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình.

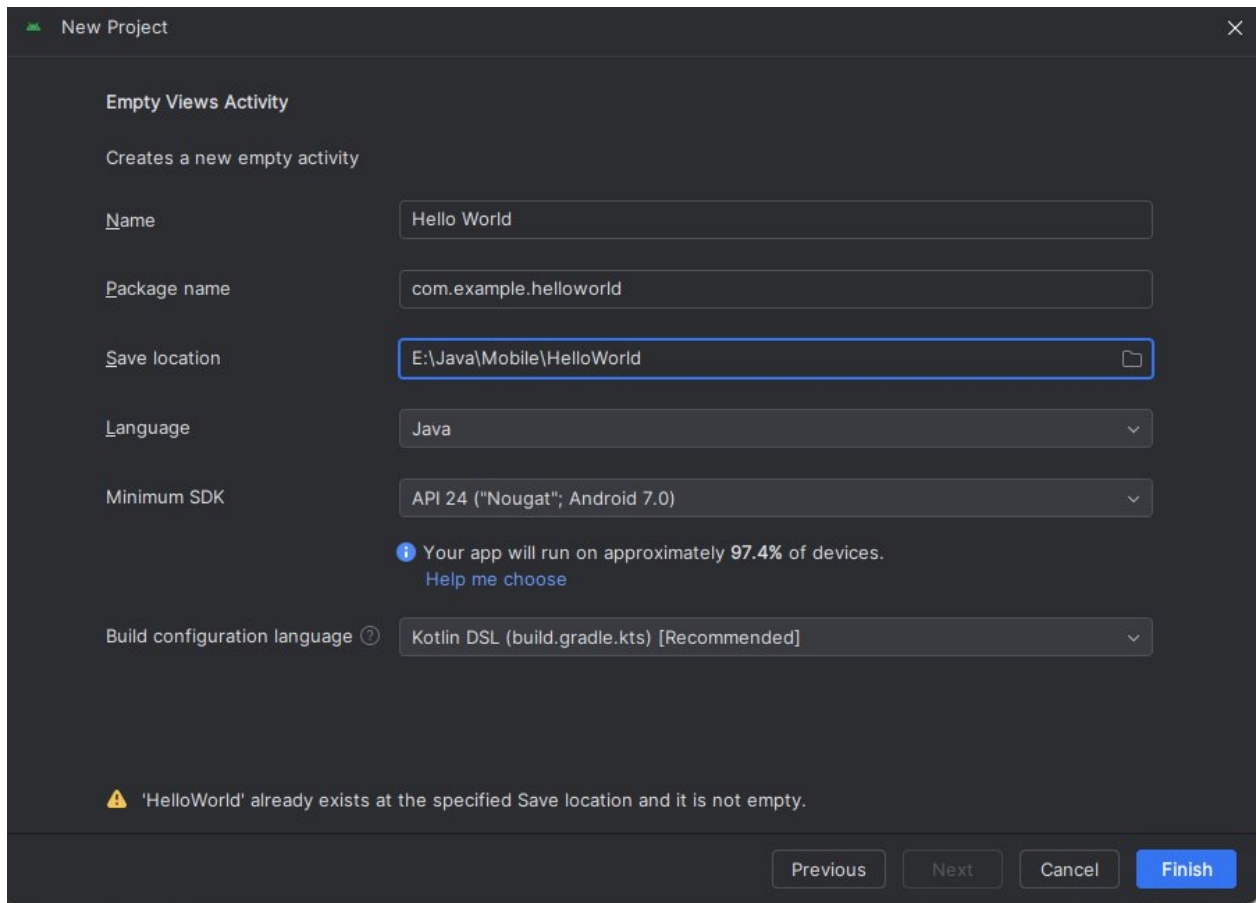
**Xử lý sự cố:** Nếu gặp vấn đề trong quá trình cài đặt, hãy kiểm tra ghi chú phát hành của Android Studio hoặc nhờ sự trợ giúp từ giảng viên của bạn.

## **Bước 2 : Tạo ứng dụng Hello World**

Trong bước này, bạn sẽ tạo một ứng dụng hiển thị dòng chữ **"Hello World"** để kiểm tra xem Android Studio đã được cài đặt đúng cách chưa, đồng thời làm quen với các bước phát triển cơ bản trong Android Studio.

### **2.1 Tạo dự án ứng dụng**

1. Mở Android Studio nếu chưa mở.
2. Trong cửa sổ chính **Welcome to Android Studio**, nhấn vào **Start a new Android Studio project**.
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** vào ô **Application name**.



4. Kiểm tra **vị trí lưu dự án** mặc định. Nếu cần, thay đổi sang thư mục mà bạn muốn lưu ứng dụng **Hello World** và các dự án khác.

5. Chấp nhận giá trị mặc định **android.example.com** cho **Company Domain**, hoặc nhập một domain riêng nếu muốn.

Nếu bạn không có ý định xuất bản ứng dụng, có thể giữ nguyên mặc định. Lưu ý rằng nếu bạn thay đổi **package name** của ứng dụng sau này, bạn sẽ phải thực hiện thêm một số bước bổ sung.

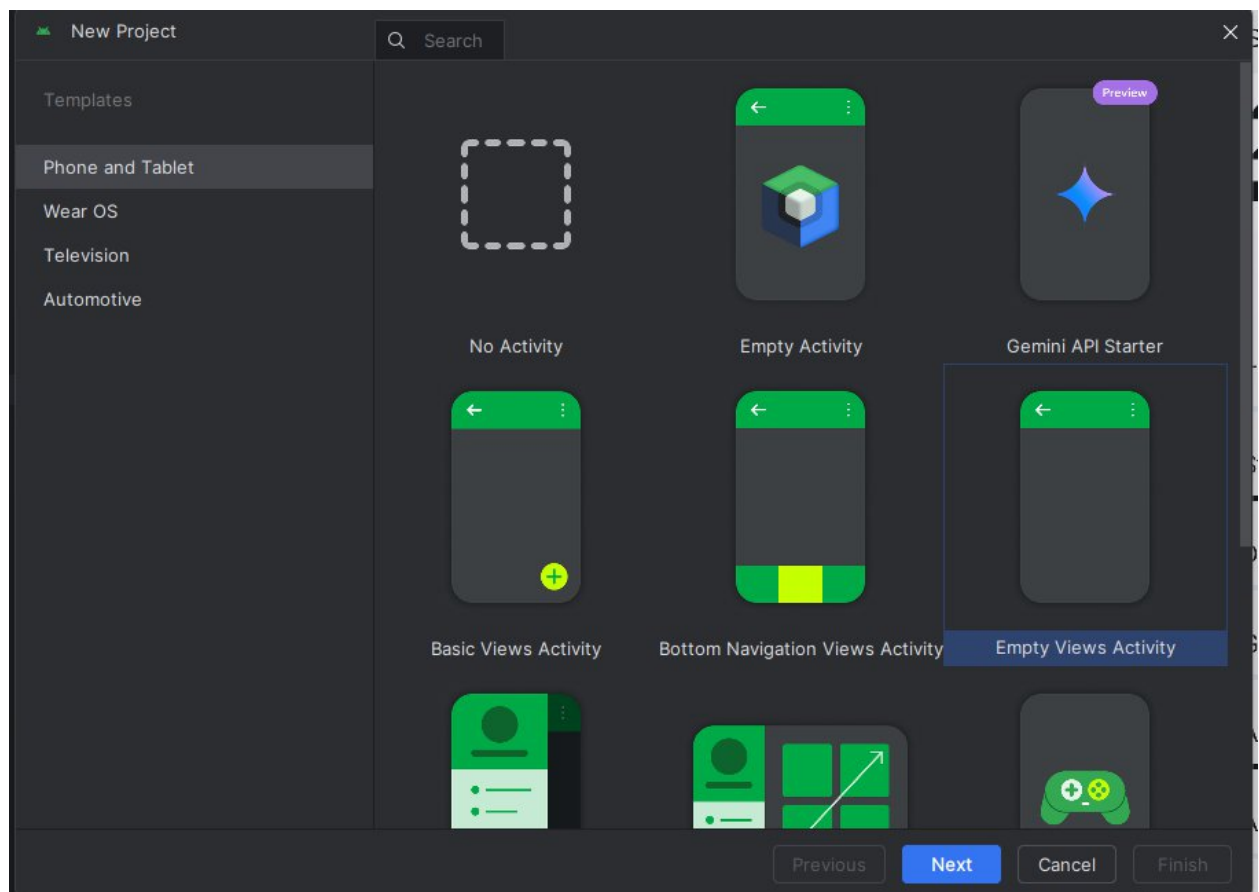
6. **Bỏ chọn** các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấn **Next**.

7. Ở màn hình **Target Android Devices**, đảm bảo **Phone and Tablet** được chọn.

Đặt **API 15: Android 4.0.3 IceCreamSandwich** làm **Minimum SDK**. Nếu API không đúng, hãy dùng menu thả xuống để chọn API phù hợp.

Các bài học trong khóa học này sử dụng các thiết lập này để đảm bảo ứng dụng **Hello World** tương thích với **97% thiết bị Android** trên Google Play Store.

8. Bỏ chọn **Include Instant App support** và tắt cả các tùy chọn khác, sau đó nhấn **Next**. Nếu dự án yêu cầu các thành phần bổ sung cho SDK bạn đã chọn, Android Studio sẽ tự động tải xuống và cài đặt chúng.
9. Ở màn hình **Add an Activity**, chọn **Empty Activity**, rồi nhấn **Next**. **Activity** là một màn hình trong ứng dụng Android, thường có một **layout** đi kèm để hiển thị giao diện người dùng. Android Studio cung cấp các mẫu Activity giúp bạn khởi tạo nhanh dự án. Với dự án Hello World, chọn **Empty Activity** như bên dưới và ấn **Next**.



10. Màn hình **Configure Activity** xuất hiện (giao diện này khác nhau tùy thuộc vào mẫu bạn đã chọn ở bước trước). Mặc định, Activity trống được cung cấp bởi mẫu có tên là **MainActivity**. Bạn có thể thay đổi nếu muốn, nhưng bài học này sử dụng **MainActivity**.



11. Đảm bảo **Generate Layout file** được chọn. Tên layout mặc định là **activity\_main**, bạn có thể đổi nếu cần, nhưng bài học này sẽ sử dụng **activity\_main**.
12. Đảm bảo **Backwards Compatibility (App Compat)** được chọn để ứng dụng tương thích với các phiên bản Android cũ hơn.
13. Nhấn **Finish**.

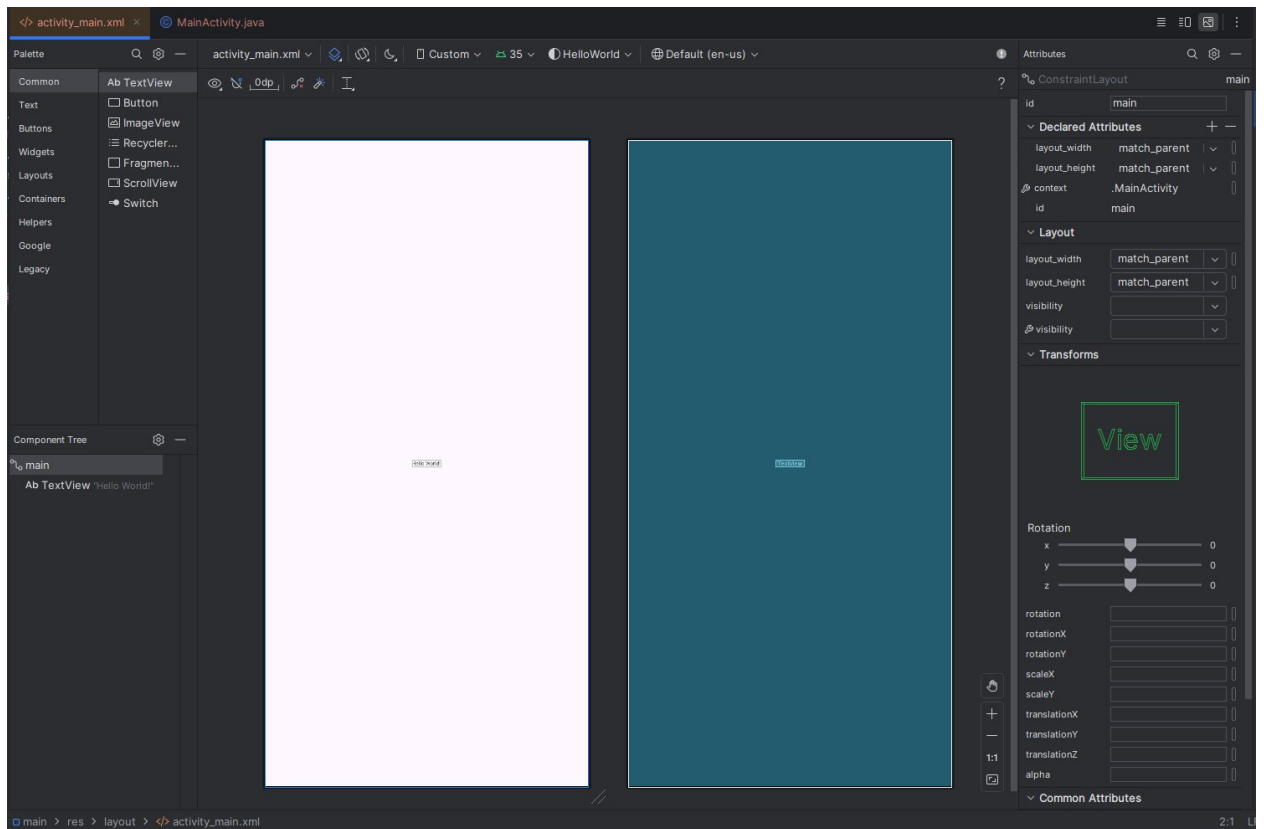
Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án với Gradle (quá trình này có thể mất một vài phút).

**Mẹo :** Xem trang dành cho nhà phát triển **Configure your build** để biết thông tin chi tiết.

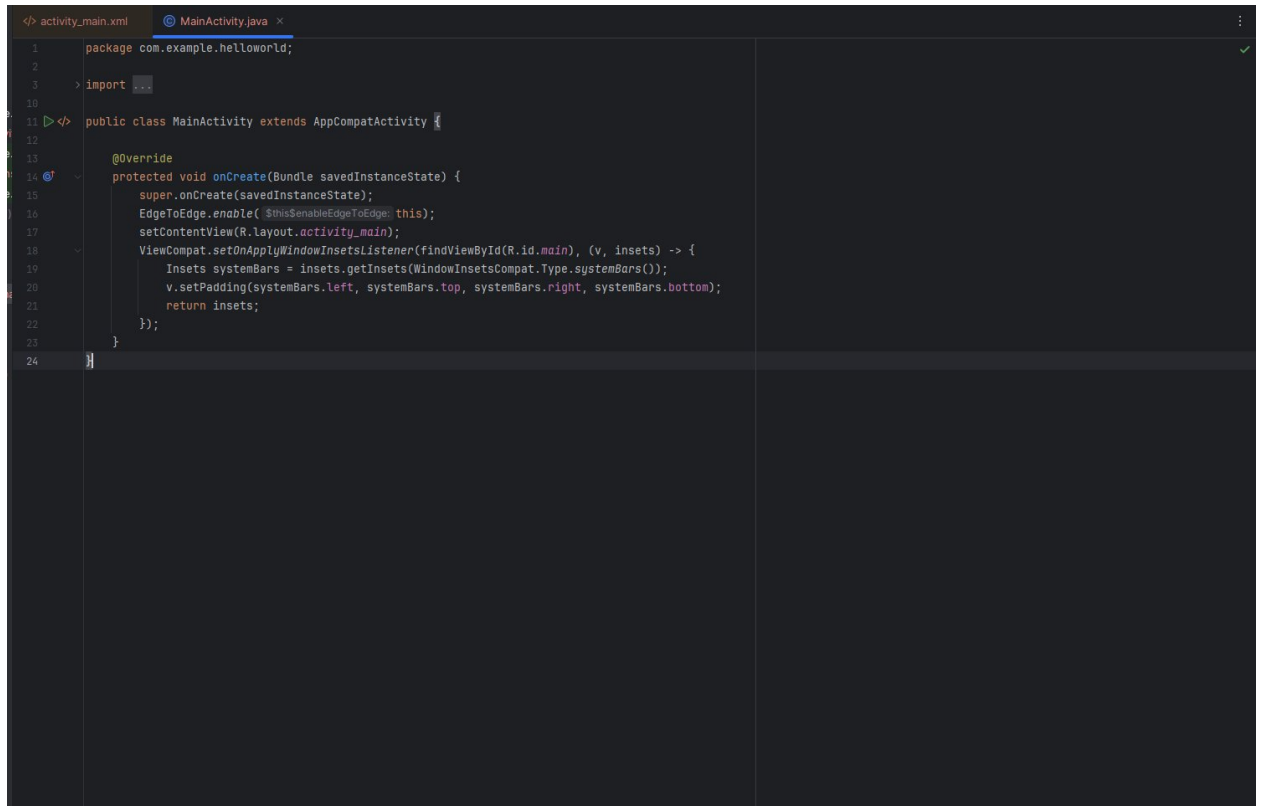
Bạn cũng có thể thấy thông báo **"Tip of the day"**, chứa các phím tắt và mẹo hữu ích khác. Nhấn **Close** để đóng thông báo.

Trình chỉnh sửa Android Studio xuất hiện. Thực hiện các bước sau:

1. Nhấp vào tab **activity\_main.xml** để mở trình chỉnh sửa giao diện.
2. Nhấp vào tab **Design** trong trình chỉnh sửa giao diện (nếu chưa được chọn) để hiển thị bản xem trước đồ họa của bố cục như hình dưới đây.



3. Nhấp vào tab **MainActivity.java** để xem trình chỉnh sửa mã như hình dưới đây.

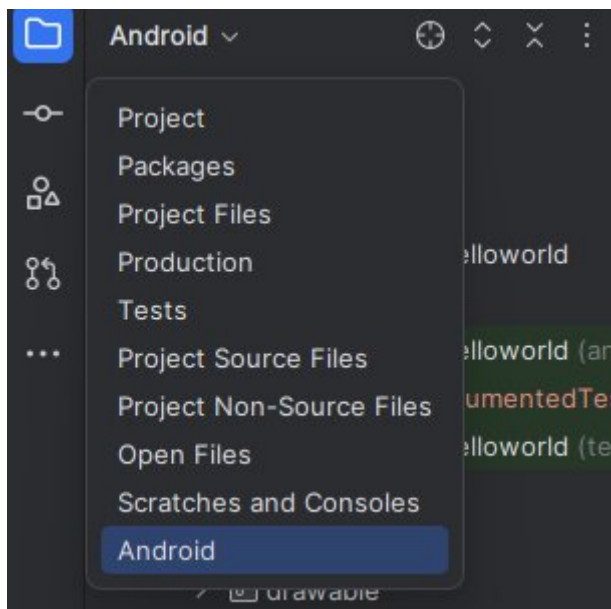
A screenshot of the Android Studio code editor. The top bar shows two tabs: 'activity\_main.xml' and 'MainActivity.java'. The 'MainActivity.java' tab is active. The code is written in Java and defines a class 'MainActivity' that extends 'AppCompatActivity'. The class has an '@Override' annotation and a 'protected void onCreate(Bundle savedInstanceState)' method. Inside the 'onCreate' method, it calls 'super.onCreate(savedInstanceState)', enables 'EdgeToEdge' with 'EdgeToEdge.enable(this)', sets the content view to 'R.layout.activity\_main', and sets a 'ViewCompat.setOnApplyWindowInsetsListener' for 'R.id.main'. The listener is a lambda function that gets 'systemBars' from 'insets.getInsets(WindowInsetsCompat.Type.systemBars())', sets padding on the view 'v' for 'systemBars.left', 'systemBars.top', 'systemBars.right', and 'systemBars.bottom', and returns 'insets'. The code is line-numbered from 1 to 24. The background is dark, and the text is light gray.

```
1 package com.example.helloworld;
2
3 > import ...
10
11 public class MainActivity extends AppCompatActivity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         EdgeToEdge.enable(this);
17         setContentView(R.layout.activity_main);
18         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
19             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
20             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
21             return insets;
22         });
23     }
24 }
```

## 2.2 Khám phá Project > Android pane

Trong bài thực hành này, bạn sẽ khám phá cách tổ chức dự án trong Android Studio.

1. Nếu chưa được chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở bên trái cửa sổ Android Studio. Pane **Project** sẽ xuất hiện.
2. Để xem dự án theo hệ thống thư mục tiêu chuẩn của Android, chọn **Android** từ menu thả xuống ở đầu pane **Project**, như hình dưới đây.

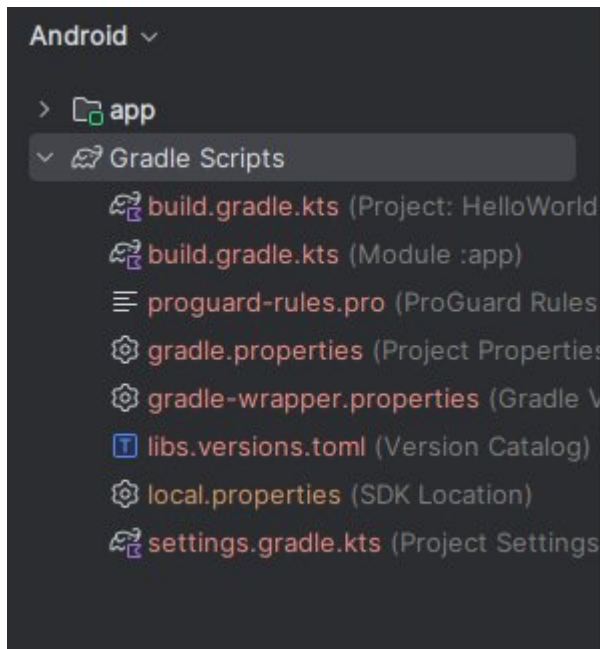


**Lưu ý:** Chương này và các chương khác đề cập đến Project pane, khi được đặt thành **Android**, là **Project > Android** pane.

## 2.3 Khám phá thư mục Gradle Scripts

Hệ thống build Gradle trong Android Studio giúp dễ dàng thêm các tệp nhị phân bên ngoài hoặc các module thư viện khác vào bản build của bạn dưới dạng dependencies.

Khi bạn tạo một dự án ứng dụng lần đầu tiên, **Project > Android** pane sẽ xuất hiện với thư mục **Gradle Scripts** được mở rộng như hình dưới đây.



Thực hiện các bước sau để khám phá hệ thống Gradle:

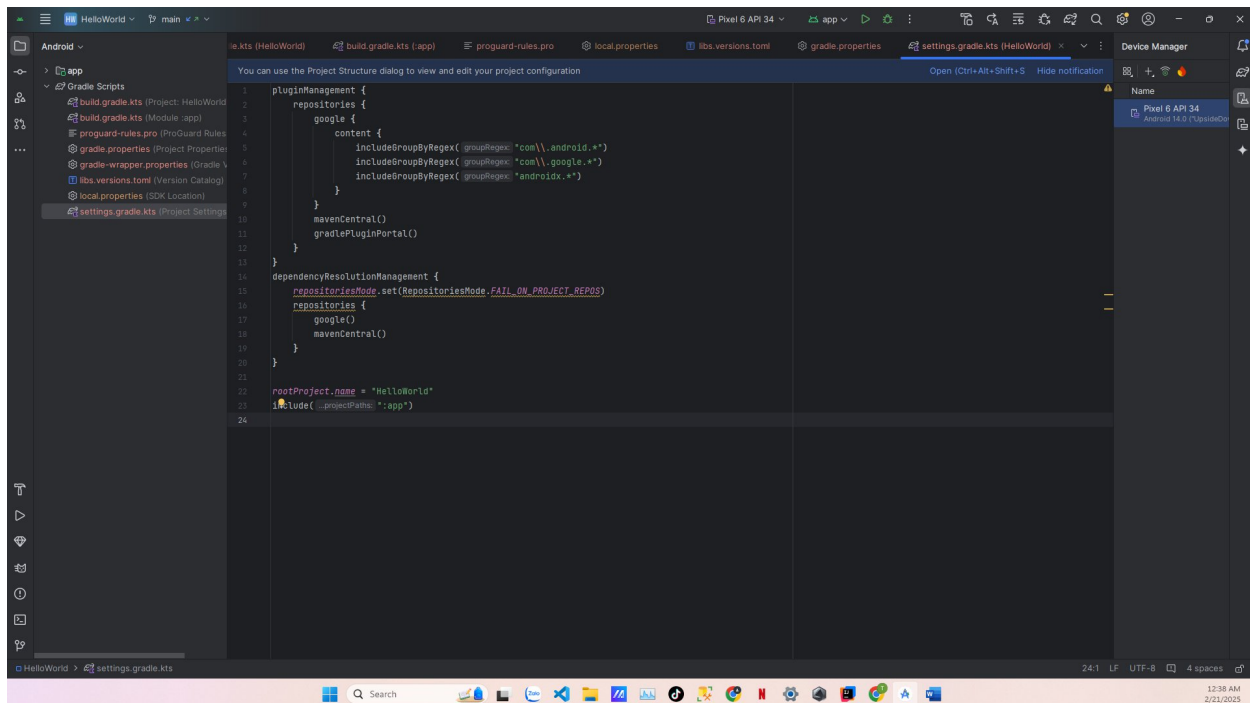
1. Nếu thư mục **Gradle Scripts** chưa được mở rộng, hãy nhấp vào biểu tượng tam giác để mở rộng.  
Thư mục này chứa tất cả các tệp cần thiết cho quá trình biên dịch (build) của dự án.

2. Tìm tệp **build.gradle (Project: HelloWorld)**.

Đây là nơi chứa các tùy chọn cấu hình chung cho tất cả các mô-đun trong dự án của bạn. Mọi dự án trong Android Studio đều có một tệp Gradle build cấp cao nhất. Thông thường, bạn sẽ không cần chỉnh sửa tệp này, nhưng việc hiểu nội dung của nó vẫn rất quan trọng.

Theo mặc định, tệp build cấp cao nhất sử dụng khối **buildscript** để khai báo kho lưu trữ Gradle và các thư viện phụ thuộc chung cho toàn bộ dự án. Khi dự án cần một thư viện không phải là thư viện cục bộ hoặc tập tin trong máy, Gradle sẽ tìm kiếm thư viện đó trong các kho lưu trữ trực tuyến được xác định trong khối **repositories** của tệp này.

Mặc định, các dự án mới trong Android Studio sử dụng **JCenter** và **Google** (bao gồm kho Maven của Google) làm kho lưu trữ chính :

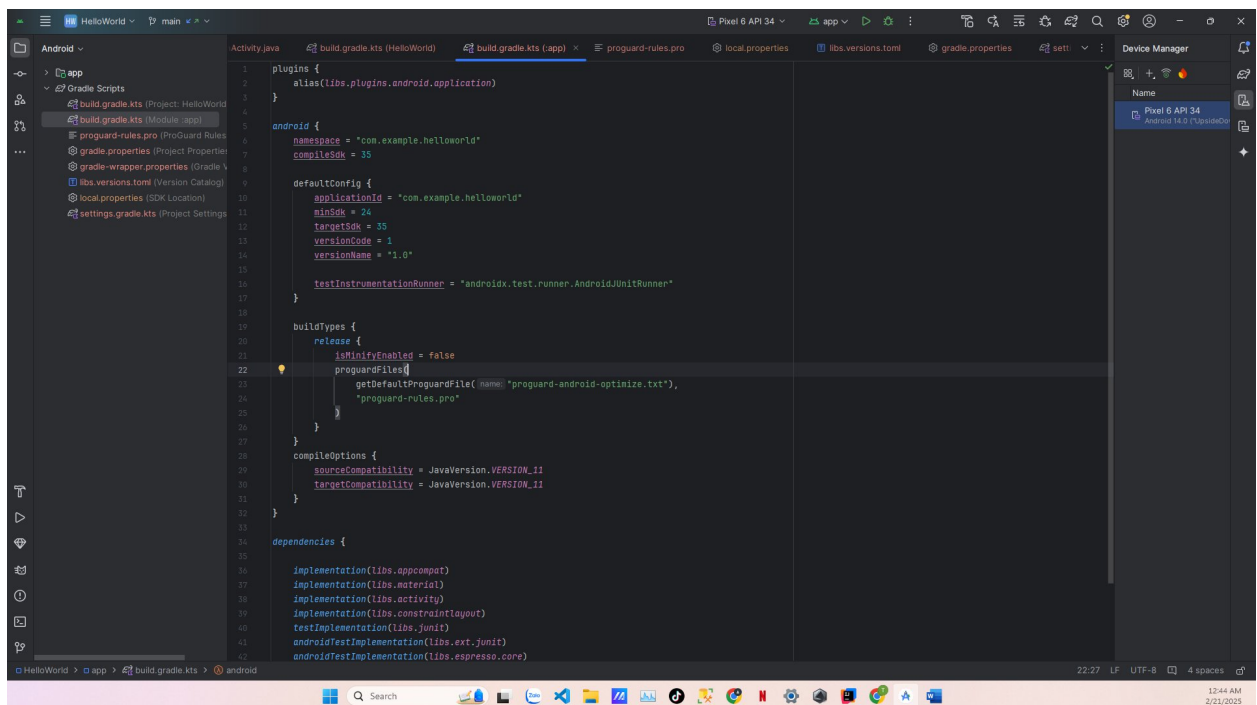


### 3. Tìm tệp **build.gradle(Module: app)**.

Ngoài tệp **build.gradle** cấp dự án, mỗi mô-đun đều có một tệp **build.gradle** riêng, cho phép bạn cấu hình cài đặt biên dịch cho từng mô-đun cụ thể (ứng dụng HelloWorld chỉ có một mô-đun). Việc cấu hình các cài đặt biên dịch này cho phép bạn cung cấp các tùy chọn đóng gói tùy chỉnh, chẳng hạn như các kiểu biên dịch bổ sung và các phiên bản sản phẩm khác nhau. Bạn cũng có thể ghi đè các cài đặt trong tệp **AndroidManifest.xml** hoặc tệp **build.gradle** cấp cao nhất.

Tệp này thường là tệp cần chỉnh sửa khi thay đổi cấu hình cấp ứng dụng, chẳng hạn như khai báo dependencies trong phần dependencies. Bạn có thể khai báo một thư viện phụ thuộc bằng một trong nhiều cấu hình phụ thuộc khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ, câu lệnh: "implementation fileTree(dir: 'libs', include: ['\*.jar'])" sẽ thêm tất cả các tệp ".jar" bên trong thư mục libs làm phụ thuộc.

Dưới đây là tệp **build.gradle(Module: app)** của ứng dụng **HelloWorld**:

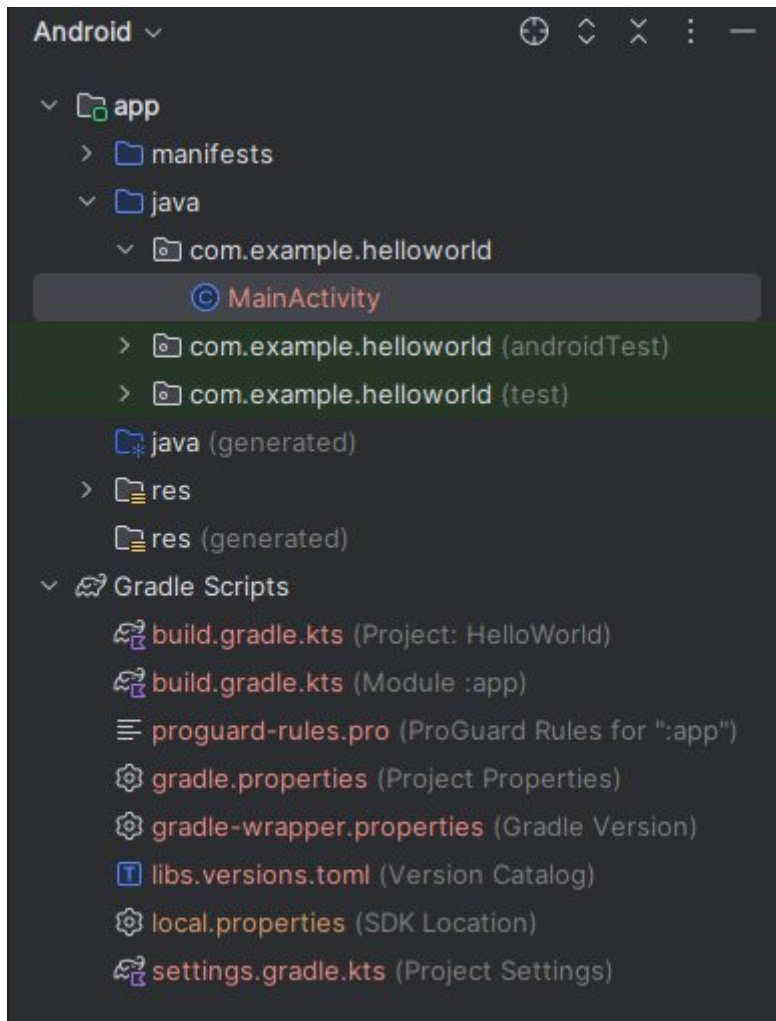


4. Nhấn vào biểu tượng tam giác để đóng thư mục **Gradle Scripts**.

## 2.4 Khám phá thư mục app và res

Tất cả mã nguồn và tài nguyên của ứng dụng nằm trong thư mục **app** và **res**.

1. Mở rộng thư mục **app**, thư mục **java**, và thư mục **com.example.android.helloworld** để xem tệp **MainActivity.java**. Nhấp đúp vào tệp để mở trong trình chỉnh sửa mã.

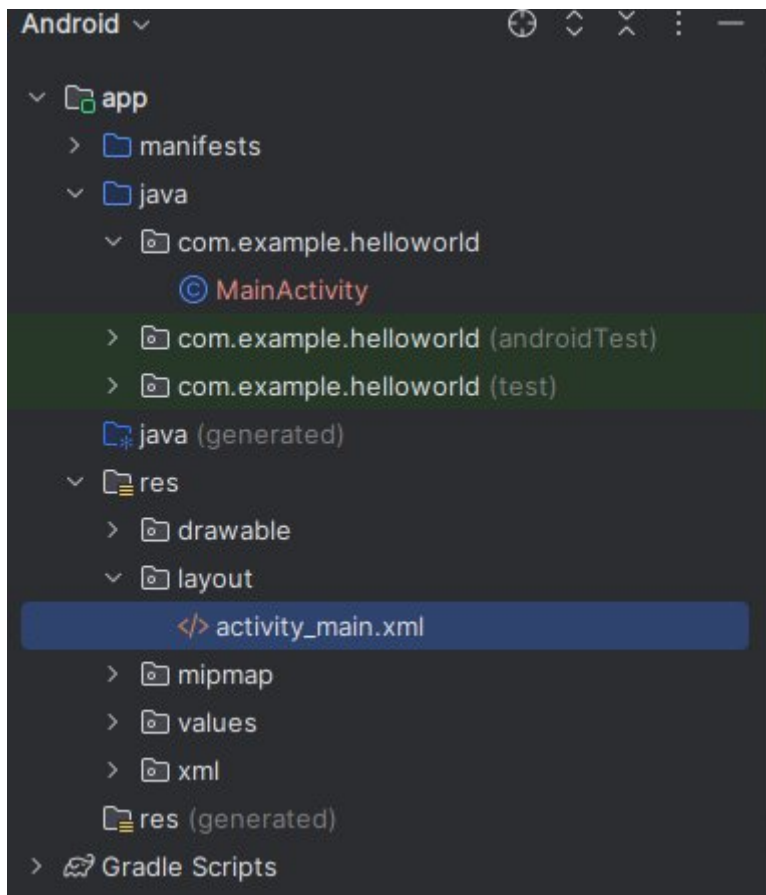


Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như trong hình trên.

Thư mục **com.example.hello.helloworld** (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp của một gói ứng dụng. Hai thư mục còn lại được sử dụng để kiểm thử và sẽ được đề cập trong bài học khác. Đối với ứng dụng **Hello World**, chỉ có một gói và nó chứa **MainActivity.java**. Tên của Activity đầu tiên mà người dùng nhìn thấy, đồng thời khởi tạo tài nguyên toàn cục của ứng dụng, thường được đặt là **MainActivity** (trong **Project > Android** pane, phần mở rộng tệp bị ẩn).

2. Mở rộng thư mục **res**, sau đó mở thư mục **layout**, và nhấp đúp vào tệp **activity\_main.xml** để mở nó trong trình chỉnh sửa giao diện.





Thư mục **res** chứa các tài nguyên như bố cục (layouts), chuỗi ký tự (strings) và hình ảnh (images). Một **Activity** thường đi kèm với một bố cục giao diện người dùng (UI views) được định nghĩa dưới dạng tệp XML. Tệp này thường được đặt tên theo **Activity** tương ứng.

## 2.5 Khám phá thư mục manifests

Thư mục manifests chứa các tệp cung cấp thông tin cần thiết về ứng dụng của bạn cho hệ thống Android, thông tin này phải có trước khi hệ thống có thể chạy bất kỳ mã nào của ứng dụng.

1. Mở rộng thư mục manifests.
2. Mở tệp AndroidManifest.xml.

Tệp AndroidManifest.xml mô tả tất cả các thành phần của ứng dụng Android. Mọi thành phần của ứng dụng, chẳng hạn như từng Activity, phải được khai báo trong tệp XML này. Trong các bài học khác của khóa học, bạn sẽ chỉnh sửa tệp này để thêm tính năng và quyền truy cập. Để tìm hiểu tổng quan, hãy xem App Manifest Overview.

## Bước 3 : Sử dụng trình giả lập

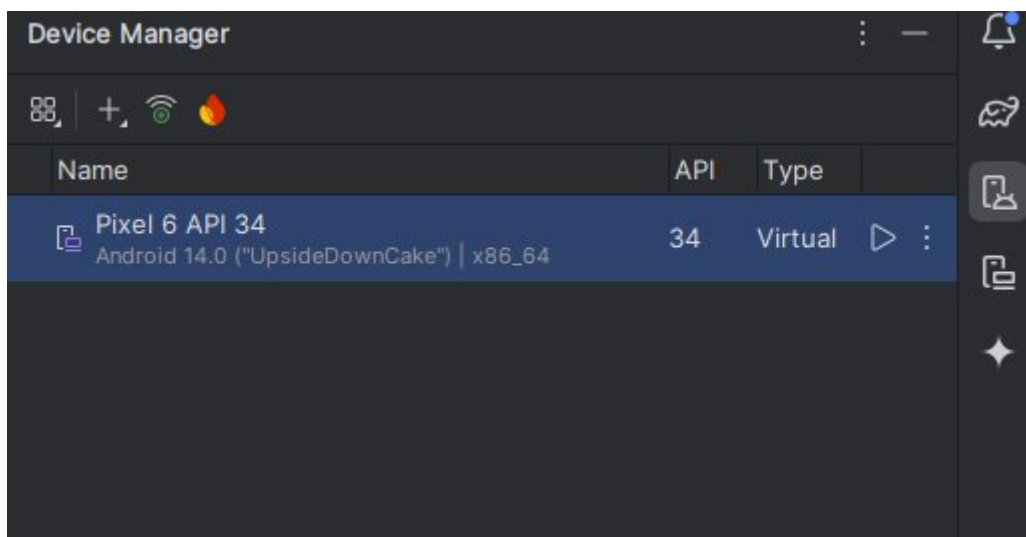
Trong nhiệm vụ này, bạn sẽ sử dụng trình quản lý Android Virtual Device (AVD) để tạo một thiết bị ảo (còn gọi là trình giả lập) mô phỏng cấu hình của một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng Android Emulator có các yêu cầu bổ sung ngoài các yêu cầu hệ thống cơ bản cho Android Studio.

Bằng cách sử dụng AVD Manager, bạn có thể xác định các đặc điểm phần cứng của thiết bị, mức API, bộ nhớ, giao diện và các thuộc tính khác, sau đó lưu chúng dưới dạng một thiết bị ảo. Với các thiết bị ảo, bạn có thể kiểm thử ứng dụng trên nhiều cấu hình thiết bị khác nhau (chẳng hạn như máy tính bảng và điện thoại) với các mức API khác nhau mà không cần sử dụng thiết bị vật lý.

### 3.1 Tạo một thiết bị ảo Android (AVD)

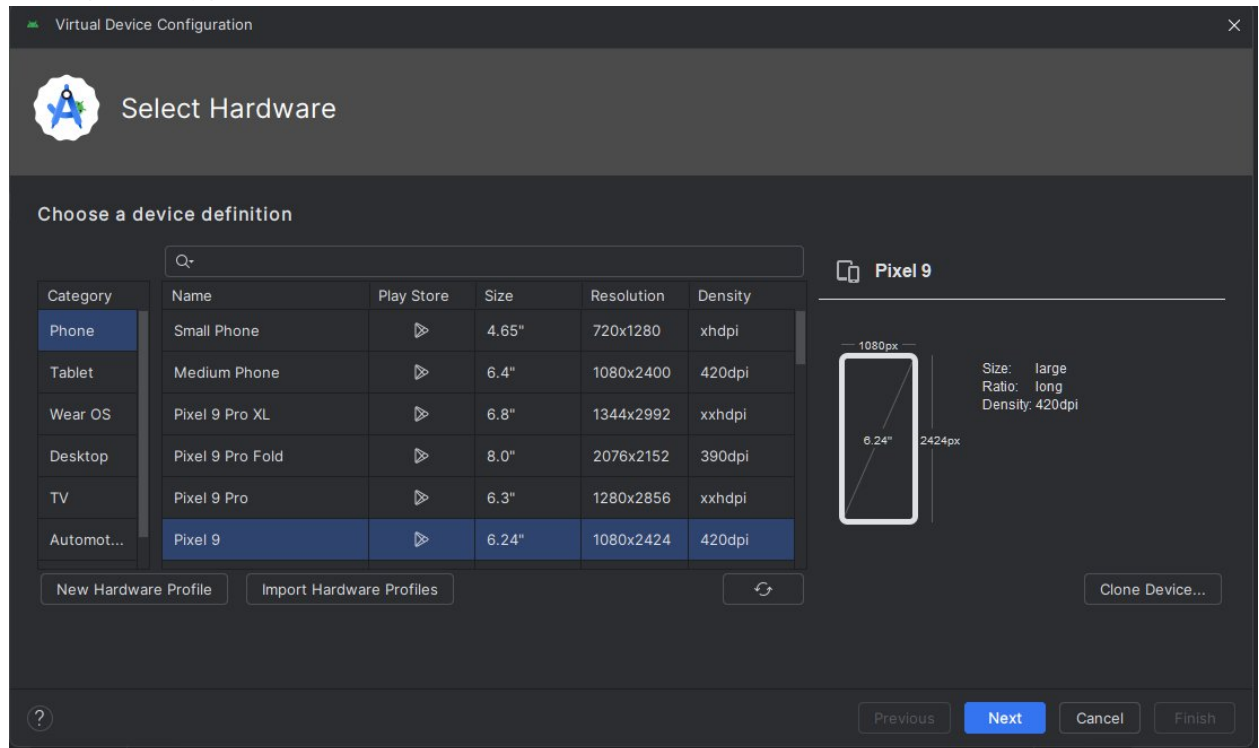
Để chạy trình giả lập trên máy tính, bạn cần tạo một cấu hình mô tả thiết bị ảo.

1. Trong Android Studio, chọn **Tools > Android > AVD Manager**, hoặc nhấp vào biểu tượng **AVD Manager** trên thanh công cụ. Màn hình **Your Virtual Devices** xuất hiện. Nếu bạn đã tạo các thiết bị ảo trước đó, chúng sẽ được hiển thị trên màn hình (như trong hình minh họa); nếu chưa, danh sách sẽ trống.

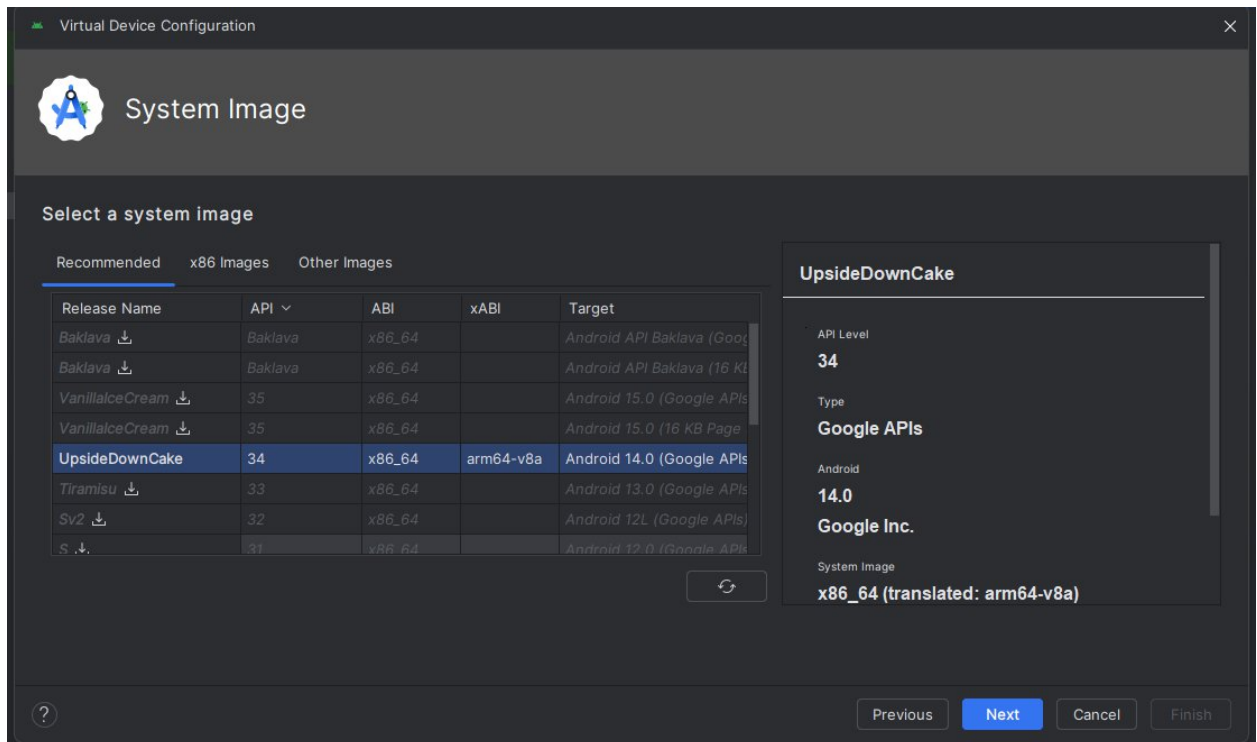


2. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng cung cấp các cột thông tin bao gồm: **Kích thước đường chéo màn**

hình (*Size*), Độ phân giải màn hình (pixel) (*Resolution*), và Mật độ điểm ảnh (*Density*).



3. Chọn một thiết bị như **Nexus 5X** hoặc **Pixel XL**, và nhấp vào **Next**. Màn hình **System Image** xuất hiện.
4. Nhấp vào tab **Recommended** nếu nó chưa được chọn, và chọn phiên bản hệ điều hành Android để chạy trên thiết bị ảo (ví dụ: **Oreo**).



Có nhiều phiên bản khác ngoài những phiên bản hiển thị trong tab **Recommended**. Hãy xem các tab **x86 Images** và **Other Images** để tìm thêm.

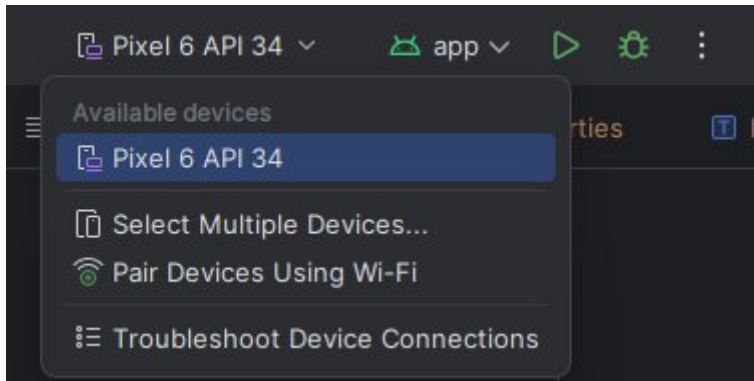
Nếu có liên kết **Download** bên cạnh một system image bạn muốn sử dụng, điều đó có nghĩa là nó chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống và nhấp vào **Finish** khi hoàn tất.

- Sau khi chọn system image, nhấp vào **Next**. Màn hình **Android Virtual Device (AVD)** xuất hiện. Bạn cũng có thể thay đổi tên của AVD. Kiểm tra cấu hình và nhấp vào **Finish**.

### 3.2 Chạy ứng dụng trên thiết bị ảo

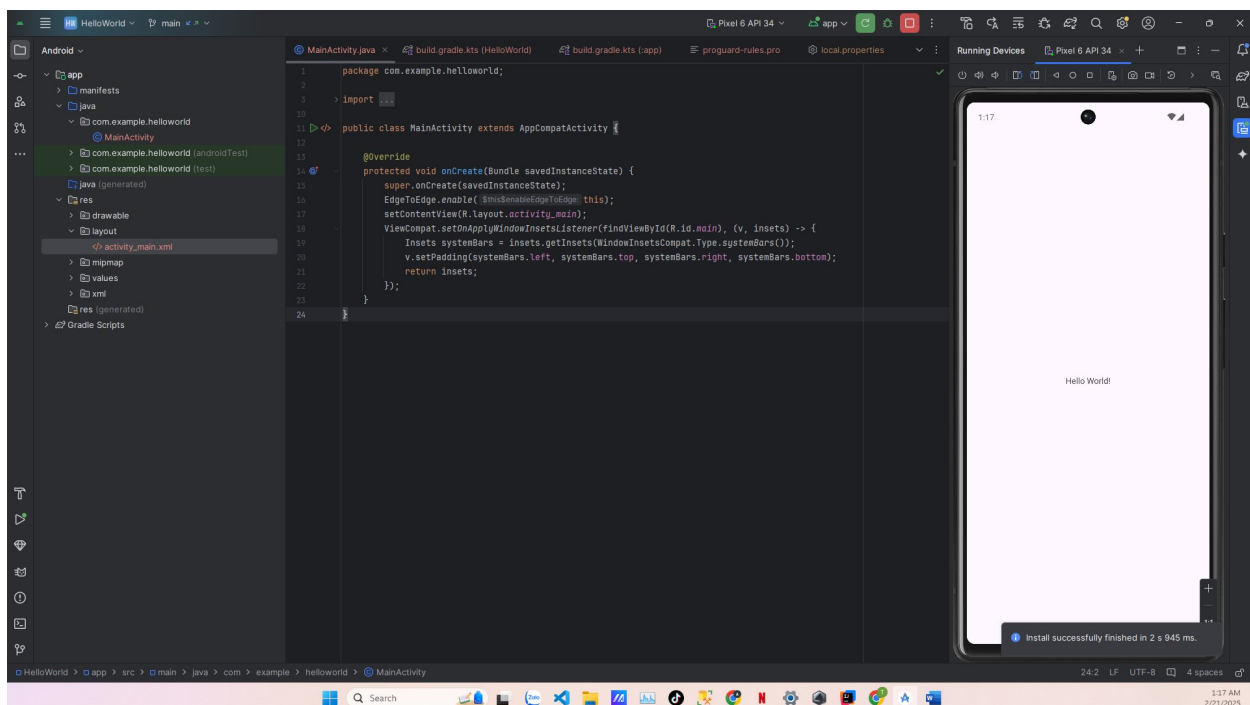
Trong nhiệm vụ này, bạn sẽ hoàn thành chạy ứng dụng Hello World của bạn.

- Trong Android Studio, chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run** trên thanh công cụ.
- Trong cửa sổ **Select Deployment Target**, dưới **Available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo và nhấp vào **OK**.



Trình giả lập khởi động và chạy giống như một thiết bị vật lý. Tùy thuộc vào tốc độ của máy tính, quá trình này có thể mất một khoảng thời gian. Ứng dụng của bạn sẽ được biên dịch, và khi trình giả lập sẵn sàng, Android Studio sẽ tải ứng dụng lên trình giả lập và chạy nó.

Bạn sẽ thấy ứng dụng **Hello World** hiển thị như trong hình minh họa sau.



**Mẹo:** Khi kiểm thử trên thiết bị ảo, bạn nên khởi động nó một lần ngay từ đầu phiên làm việc. Không nên đóng trình giả lập cho đến khi hoàn tất kiểm thử ứng dụng để tránh phải khởi động lại thiết bị. Để đóng thiết bị ảo, nhấn nút **X** ở góc trên của trình giả lập, chọn **Quit** từ menu hoặc nhấn **Control-Q** trên Windows hoặc **Command-Q** trên macOS.

## Bước 4 : (Tùy chọn) Sử dụng thiết bị thật

Trong nhiệm vụ cuối cùng này, bạn sẽ chạy ứng dụng trên một thiết bị di động vật lý như điện thoại hoặc máy tính bảng. Bạn nên luôn kiểm thử ứng dụng trên cả thiết bị ảo và thiết bị vật lý.

Những gì bạn cần:

- Một thiết bị Android như điện thoại hoặc máy tính bảng.
- Cáp dữ liệu để kết nối thiết bị Android với máy tính qua cổng USB.
- Nếu bạn sử dụng hệ điều hành Linux hoặc Windows, có thể cần thực hiện thêm một số bước để chạy ứng dụng trên thiết bị phần cứng. Hãy kiểm tra tài liệu Using Hardware Devices. Bạn cũng có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Đối với trình điều khiển USB trên Windows, hãy xem OEM USB Drivers.

## 4.1 Bật USB debugging

Để Android Studio có thể giao tiếp với thiết bị của bạn, bạn cần bật **USB Debugging** trên thiết bị Android. Tùy chọn này được kích hoạt trong **Developer options** của thiết bị.

Trên Android 4.2 trở lên, màn hình **Developer options** bị ẩn theo mặc định. Để hiển thị và bật **USB Debugging**, hãy làm theo các bước sau:

1. Trên thiết bị, mở **Cài đặt (Settings)**, tìm **Giới thiệu về điện thoại (About phone)**, nhấn vào đó và chạm vào **Số bản dựng (Build number)** bảy lần.
2. Quay lại màn hình trước (**Cài đặt / Hệ thống - Settings / System**). Tùy chọn **Developer options** sẽ xuất hiện trong danh sách. Nhấn vào **Developer options**.
3. Bật **USB Debugging**.

## 4.2 Chạy ứng dụng của bạn trên thiết bị

Bây giờ bạn có thể kết nối thiết bị và chạy ứng dụng từ Android Studio.

1. Kết nối thiết bị với máy tính bằng cáp USB.
2. Nhấn nút **Run** trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ mở ra, hiển thị danh sách các trình giả lập và thiết bị được kết nối.
3. Chọn thiết bị của bạn và nhấn **OK**.

Android Studio sẽ cài đặt và chạy ứng dụng trên thiết bị của bạn.

## Khắc phục sự cố

Nếu Android Studio không nhận diện được thiết bị của bạn, hãy thử các bước sau:

1. Rút và cắm lại thiết bị.
2. Khởi động lại Android Studio.

Nếu máy tính vẫn không tìm thấy thiết bị hoặc báo lỗi "unauthorized", hãy làm theo các bước sau:

1. Rút kết nối thiết bị.
2. Trên thiết bị, mở **Developer Options** trong ứng dụng **Settings**.
3. Nhấn **Revoke USB Debugging authorizations**.
4. Kết nối lại thiết bị với máy tính.
5. Khi được nhắc, cấp quyền xác thực.

Bạn có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Xem tài liệu **Using Hardware Devices** để biết thêm chi tiết.

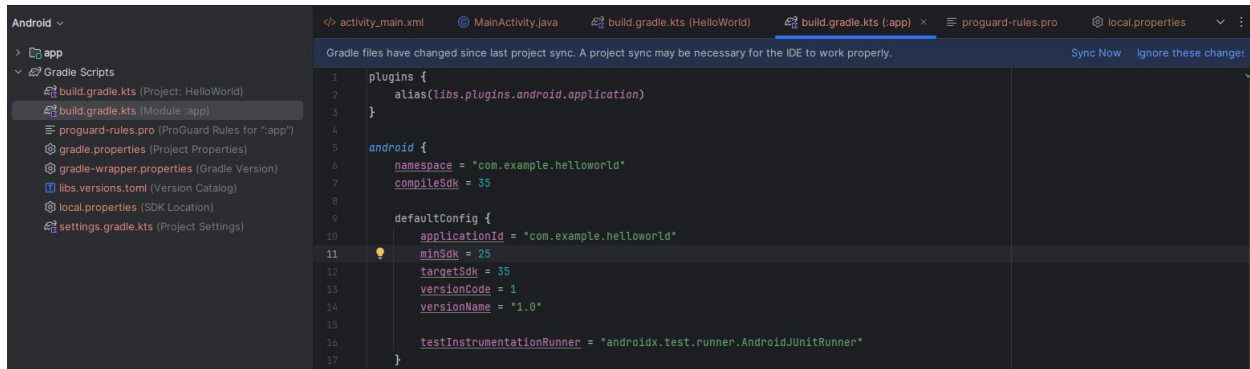
## Bước 5: Thay đổi cấu hình Gradle của ứng dụng

Trong nhiệm vụ này, bạn sẽ thay đổi một số cấu hình của ứng dụng trong tệp **build.gradle(Module:app)** để tìm hiểu cách thực hiện thay đổi và đồng bộ chúng với dự án Android Studio của bạn.

### 5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện theo các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa được mở và nhấp đúp vào tệp **build.gradle(Module:app)**. Nội dung của tệp sẽ xuất hiện trong trình chỉnh sửa mã.
2. Trong khối **defaultConfig**, thay đổi giá trị của **minSdkVersion** thành **17** như bên dưới (giá trị ban đầu là 15).



Trình chỉnh sửa mã hiển thị một thanh thông báo ở trên cùng với liên kết **Sync Now**.

## 5.2 Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi trong các tệp cấu hình build của dự án, Android Studio yêu cầu bạn đồng bộ hóa các tệp dự án để có thể nhập các thay đổi cấu hình build và chạy một số kiểm tra nhằm đảm bảo cấu hình không gây ra lỗi build.

Để đồng bộ các tệp dự án, nhấp vào **Sync Now** trong thanh thông báo xuất hiện khi bạn thực hiện thay đổi (như trong hình trước đó) hoặc nhấp vào biểu tượng **Sync Project with Gradle Files** trên thanh công cụ.

Khi quá trình đồng bộ Gradle hoàn tất, thông báo **Gradle build finished** sẽ xuất hiện ở góc dưới bên trái cửa sổ Android Studio.

Để tìm hiểu sâu hơn về Gradle, hãy tham khảo tài liệu **Build System Overview** và **Configuring Gradle Builds**.

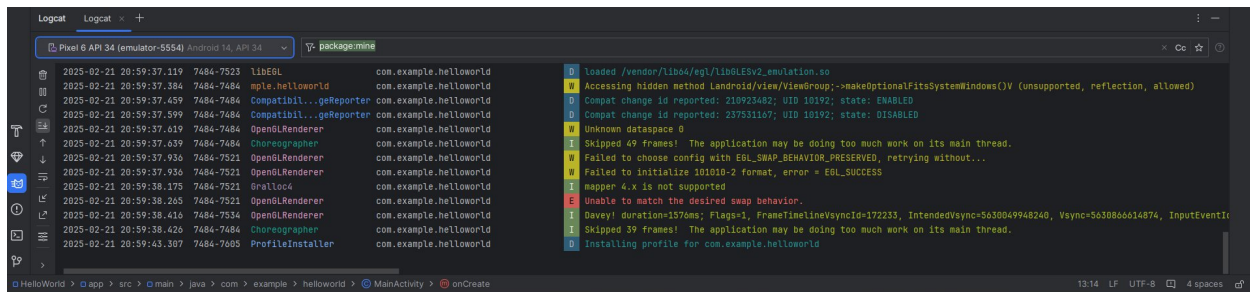
## Bước 6: Thêm các câu lệnh log vào ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh Log vào ứng dụng của mình, hiển thị thông báo trong bảng Logcat. Thông báo Log là một công cụ gỡ lỗi mạnh mẽ mà bạn có thể sử dụng để kiểm tra giá trị, luồng thực thi và báo cáo ngoại lệ.

### 6.1 Xem bảng Logcat

Để xem bảng Logcat, hãy nhấp vào tab Logcat ở phía dưới cửa sổ Android Studio, như minh họa trong hình dưới đây.





Trong hình trên:

1. Tab **Logcat** để mở và đóng bảng Logcat, hiển thị thông tin về ứng dụng khi đang chạy. Nếu bạn thêm các câu lệnh Log vào ứng dụng, thông báo Log sẽ xuất hiện ở đây.
2. Menu mức Log được đặt thành **Verbose** (mặc định), hiển thị tất cả thông báo Log. Các cài đặt khác bao gồm **Debug**, **Error**, **Info** và **Warn**.

## 6.2 Thêm câu lệnh Log vào ứng dụng của bạn

Các câu lệnh Log trong mã ứng dụng của bạn hiển thị thông báo trong bảng Logcat. Ví dụ: **Log.d("MainActivity", "Hello World");**

Các phần của thông báo gồm:

- **Log**: Lớp Log dùng để gửi thông báo nhật ký đến bảng Logcat.
- **d**: Mức nhật ký Debug để lọc hiển thị thông báo trong bảng Logcat. Các mức nhật ký khác bao gồm **e** cho Error, **w** cho Warn và **i** cho Info.
- **"MainActivity"**: Đối số đầu tiên là một thẻ (tag) có thể được sử dụng để lọc thông báo trong bảng Logcat. Thông thường, đây là tên của **Activity** nơi thông báo được tạo ra. Tuy nhiên, bạn có thể đặt bất kỳ giá trị nào hữu ích cho quá trình gỡ lỗi.

Theo quy ước, các thẻ log được định nghĩa dưới dạng hằng số trong **Activity**:

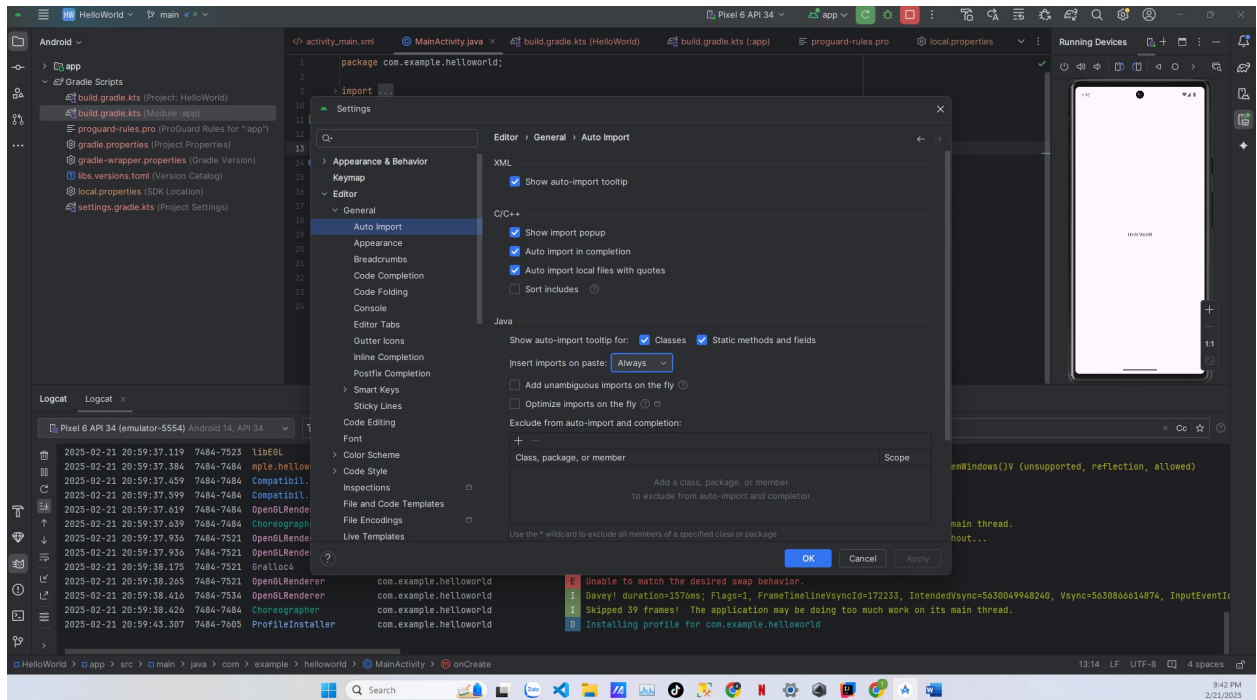
```
private static final String LOG_TAG =
MainActivity.class.getSimpleName();
```

- **"Hello world"**: Đối số thứ hai là thông báo thực tế.

Làm theo các bước sau:

1. Mở ứng dụng Hello World trong Android Studio và mở **MainActivity**.

- Để tự động thêm các import không mở hồ vào dự án của bạn (chẳng hạn như `android.util.Log` cần thiết để sử dụng `Log`), chọn **File > Settings** trong Windows hoặc **Android Studio > Preferences** trong macOS.
- Chọn **Editor > General > Auto Import**. Chọn tất cả các hộp kiểm và đặt **Insert imports on paste** thành **All**.



- Nhấp vào **Apply**, sau đó nhấp vào **OK**.
- Trong phương thức `onCreate()` của **MainActivity**, thêm câu lệnh sau:  
**Log.d("MainActivity", "Hello World");**

Phương thức `onCreate()` bây giờ sẽ trông như đoạn mã sau:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_main);
    Log.d("MainActivity", "Hello World");
}
```

- Nếu Logcat chưa mở, hãy nhấp vào tab Logcat ở cuối cửa sổ Android Studio để mở nó.
- Kiểm tra xem tên mục tiêu và tên gói của ứng dụng có đúng không.

8. Thay đổi mức Log trong Logcat thành Debug (hoặc giữ nguyên là Verbose vì có rất ít thông báo log).
9. Chạy ứng dụng của bạn.

Thông báo sau đây sẽ xuất hiện trong Logcat:

```
2025-02-21 21:53:57.899 8125-8125 MainActivity com.example.helloworld D Hello World
```

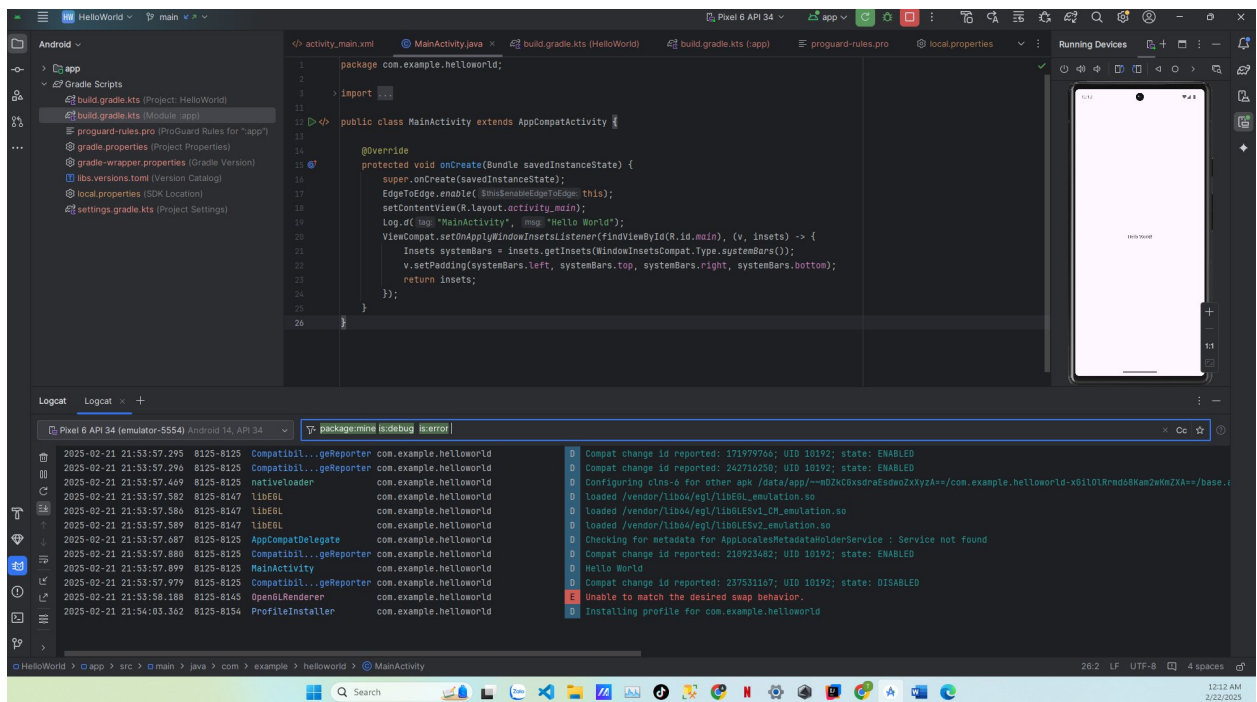
## Tổng kết

- Để cài đặt Android Studio, truy cập Android Studio và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo ứng dụng mới, đảm bảo rằng API 15: Android 4.0.3 IceCreamSandwich được đặt làm Minimum SDK.
- Để xem cấu trúc phân cấp Android của ứng dụng trong ngăn Project, nhấp vào tab Project trong cột tab dọc, sau đó chọn Android trong menu bật lên ở trên cùng.
- Chỉnh sửa tệp build.gradle(Module:app) khi cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.

## Bài tập về nhà

### Xây dựng và chạy một ứng dụng

- Tạo một dự án Android mới từ mẫu **Empty Template**.
- Thêm các câu lệnh ghi log với nhiều mức khác nhau trong phương thức onCreate() của **MainActivity**.
- Tạo một trình giả lập (emulator) cho một thiết bị, nhắm mục tiêu bất kỳ phiên bản Android nào bạn muốn, và chạy ứng dụng.
- Sử dụng bộ lọc trong **Logcat** để tìm các câu lệnh ghi log của bạn và điều chỉnh mức lọc để chỉ hiển thị các câu lệnh ghi log ở mức **debug** hoặc **error**.



## Trả lời câu hỏi

### Câu hỏi 1

Tên của tệp layout cho activity chính là gì?

- MainActivity.java
- AndroidManifest.xml
- **activity\_main.xml**
- build.gradle

## Câu hỏi 2

Tên của resource chuỗi nào xác định tên của ứng dụng?

- **app\_name**
- xmlns:app
- android:name
- applicationId

### Câu hỏi 3

Bạn sử dụng công cụ nào để tạo một trình giả lập (emulator) mới?

- Android Device Monitor
- **AVD Manager**
- SDK Manager
- Theme Editor

## Câu hỏi 4

Giả sử ứng dụng của bạn bao gồm câu lệnh ghi log sau:

**Log.i("MainActivity", "MainActivity layout is complete");**

Bạn sẽ thấy câu lệnh "MainActivity layout is complete" trong bảng Logcat nếu menu mức log được đặt thành mức nào sau đây? (Gợi ý: Có thể có nhiều đáp án đúng.)

- **Verbose**
- Debug
- **Info**
- Warn
- Error
- Assert

## Nội dung ứng dụng của bạn để chấm điểm

Hãy kiểm tra để đảm bảo ứng dụng có các yêu cầu sau:

- Một Activity hiển thị dòng chữ "Hello World" trên màn hình.
- Các câu lệnh ghi log trong phương thức onCreate() của activity chính.
- Mức log trong bảng Logcat chỉ hiển thị các câu lệnh ghi log ở mức debug hoặc error.

### 1.2) Giao diện người dùng tương tác đầu tiên

## Mở đầu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là views — mọi phần tử trên màn hình đều là một View. Lớp View đại diện cho khối xây dựng cơ bản của tất cả các thành phần giao diện người dùng và là lớp cơ sở cho các lớp cung cấp các thành phần giao diện người dùng tương tác như nút bấm, hộp kiểm và trường nhập văn bản. Các lớp View con thường được sử dụng được mô tả trong nhiều bài học bao gồm:

- TextView để hiển thị văn bản.
- EditText cho phép người dùng nhập và chỉnh sửa văn bản.
- Button và các phần tử có thể nhấp khác (chẳng hạn như RadioButton, CheckBox và Spinner) để cung cấp hành vi tương tác.
- ScrollView và RecyclerView để hiển thị các mục có thể cuộn.
- ImageView để hiển thị hình ảnh.
- ConstraintLayout và LinearLayout để chứa các phần tử View khác và định vị chúng.

Mã Java hiển thị và điều khiển giao diện người dùng (UI) được chứa trong một lớp mở rộng từ Activity. Một Activity thường được liên kết với một bố cục giao diện người dùng được định nghĩa dưới dạng tệp XML (eXtended Markup Language). Tệp XML này thường được đặt tên theo Activity của nó và xác định bố cục của các phần tử View trên màn hình.

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bố cục được định nghĩa trong tệp bố cục activity\_main.xml, trong đó có một TextView với nội dung "Hello World".

Trong các ứng dụng phức tạp hơn, một Activity có thể thực hiện các hành động để phản hồi thao tác chạm của người dùng, vẽ nội dung đồ họa hoặc yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet. Bạn sẽ tìm hiểu thêm về lớp Activity trong một bài học khác.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên — một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo một ứng dụng sử dụng mẫu "Empty Activity". Bạn cũng sẽ học cách sử dụng trình chỉnh sửa bố cục để thiết kế giao diện và chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng này để hoàn thành các bài thực hành khác trong khóa học này.

## **Những điều Bạn nên biết**

Bạn nên làm quen với:

- Cách cài đặt và mở Android Studio
- Cách tạo ứng dụng HelloWorld
- Cách chạy ứng dụng HelloWorld\

## **Những gì Bạn sẽ học**

- Cách tạo một ứng dụng có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục để thiết kế giao diện.

- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới. Hãy tham khảo bảng thuật ngữ và khái niệm để có các định nghĩa dễ hiểu.

## **Những gì bạn sẽ làm**

- Tạo một ứng dụng và thêm hai phần tử Button cùng một TextView vào bố cục.
- Điều chỉnh từng phần tử trong ConstraintLayout để ràng buộc chúng với lề và các phần tử khác.
- Thay đổi thuộc tính của các phần tử giao diện người dùng (UI).
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi được mã hóa cứng thành tài nguyên chuỗi.
- Triển khai phương thức xử lý sự kiện nhấp để hiển thị thông báo trên màn hình khi người dùng nhấn vào từng Button.

## **Tổng quan ứng dụng**

Ứng dụng HelloToast bao gồm hai phần tử Button và một TextView. Khi người dùng nhấn vào Button đầu tiên, ứng dụng sẽ hiển thị một thông báo ngắn (Toast) trên màn hình. Nhấn vào Button thứ hai sẽ tăng bộ đếm số lần nhấp hiển thị trong TextView, bắt đầu từ số không.

Dưới đây là giao diện hoàn chỉnh của ứng dụng:

## **Bước 1 : Tạo và khám phá một dự án mới**

Trong bài thực hành này, bạn sẽ thiết kế và triển khai một dự án cho ứng dụng HelloToast. Một liên kết đến mã giải pháp sẽ được cung cấp ở cuối.

### **1.1 Tạo dự án Android Studio**

Khởi động Android Studio và tạo một dự án mới với các thông số sau:

Application Name	<b>Hello Toast</b>
Company Name	<b>com.example.android</b> (or your own domain)
Phone and Tablet Minimum SDK	<b>API15: Android 4.0.3 IceCreamSandwich</b>
Template	<b>Empty Activity</b>
Generate Layout file box	Selected
Backwards Compatibility box	Selected

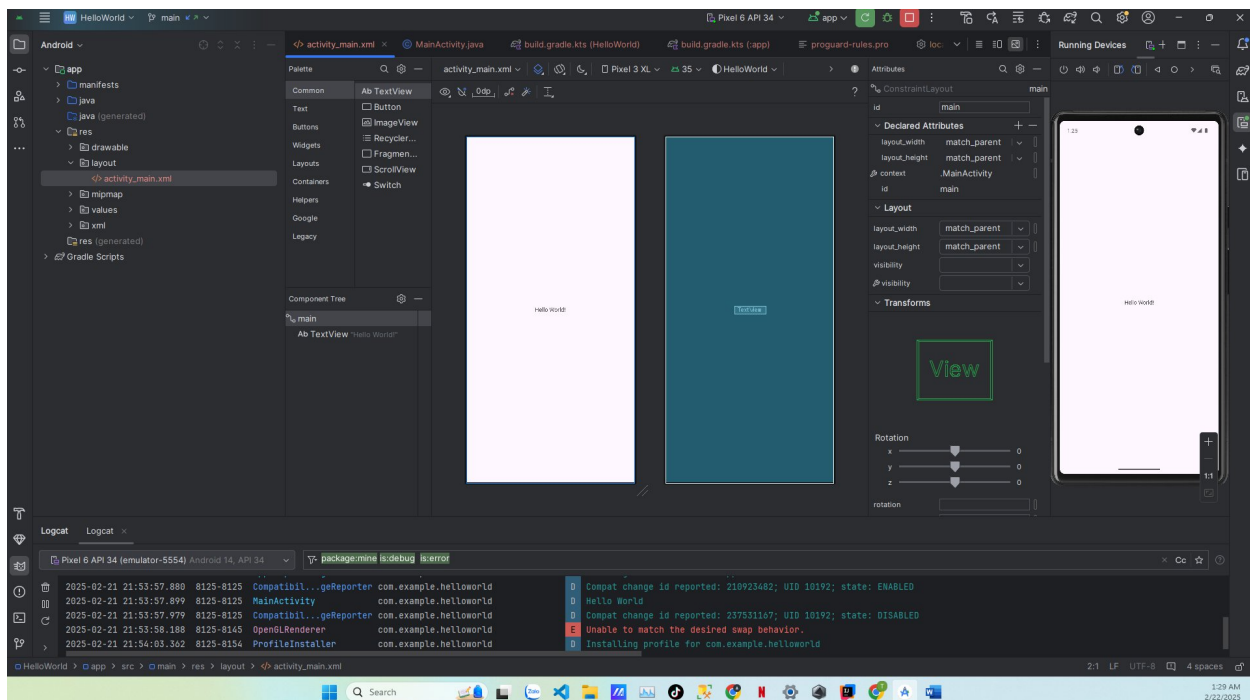
Chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run** trên thanh công cụ để biên dịch và chạy ứng dụng trên trình giả lập hoặc thiết bị của bạn.

## 1.2 Khám phá trình chỉnh sửa bố cục

Android Studio cung cấp trình chỉnh sửa bố cục (layout editor) để nhanh chóng xây dựng giao diện người dùng (UI) của ứng dụng. Nó cho phép bạn kéo các phần tử vào chế độ thiết kế trực quan và chế độ xem bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc (constraints) và đặt thuộc tính. Ràng buộc xác định vị trí của một phần tử UI trong bố cục. Một ràng buộc đại diện cho một kết nối hoặc căn chỉnh với một view khác, bố cục cha hoặc một hướng dẫn vô hình.

Khám phá trình chỉnh sửa bố cục và tham khảo hình minh họa bên dưới khi thực hiện theo các bước được đánh số.





1. Trong ngăn **Project > Android**, điều hướng đến thư mục **app > res > layout**, sau đó nhấp đúp vào tệp **activity\_main.xml** để mở, nếu nó chưa được mở.
2. Nhấp vào tab **Design** nếu nó chưa được chọn. Bạn sử dụng tab **Design** để thao tác các phần tử và bố cục, và tab **Text** để chỉnh sửa mã XML của bố cục.
3. Ngăn **Palette** hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng.
4. Ngăn **Component Tree** hiển thị cây phân cấp của các phần tử UI. Các phần tử View được tổ chức theo cấu trúc cây gồm cha và con, trong đó phần tử con kế thừa thuộc tính của phần tử cha. Trong hình minh họa, **TextView** là phần tử con của **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về các phần tử này sau trong bài học.
5. Các ngăn **Design** và **Blueprint** trong trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ chứa một phần tử: một **TextView** hiển thị dòng chữ "Hello World".
6. Tab **Attributes** hiển thị bảng **Attributes**, nơi bạn có thể đặt các thuộc tính cho một phần tử UI.

**Mẹo:** Xem **Building a UI with Layout Editor** để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục, và **Meet Android Studio** để xem toàn bộ tài liệu về Android Studio.

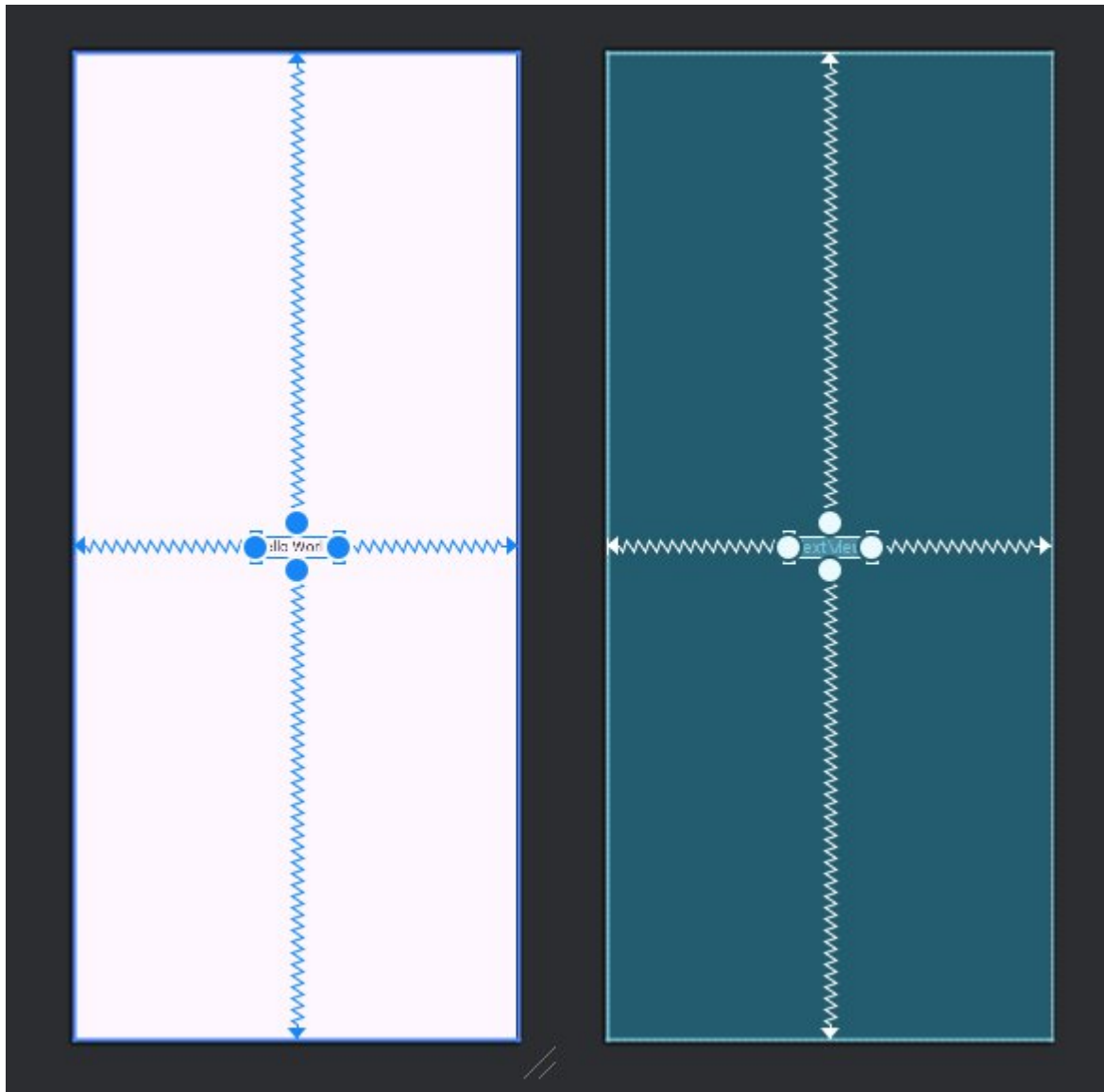
## Bước 2 : Thêm các phần tử View trong trình chỉnh sửa bố cục

Trong nhiệm vụ này, bạn sẽ tạo bố cục UI cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng cách sử dụng các tính năng của ConstraintLayout. Bạn có thể tạo các ràng buộc (constraints) theo cách thủ công, như sẽ được trình bày sau, hoặc tự động bằng công cụ Autoconnect.

### 2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện các bước sau:

1. Mở **activity\_main.xml** từ **Project > Android** pane nếu nó chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào nó.  
  
Nếu không có bản thiết kế (**blueprint**), nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**.
2. Công cụ **Autoconnect** cũng nằm trên thanh công cụ. Nó được bật theo mặc định. Đối với bước này, hãy đảm bảo công cụ không bị vô hiệu hóa.
3. Nhấp vào nút **Zoom in** để phóng to bảng thiết kế và bản thiết kế để xem cận cảnh.
4. Chọn **TextView** trong bảng **Component Tree**. **TextView** hiển thị "Hello World" sẽ được tô sáng trong bảng thiết kế và bản thiết kế, đồng thời các ràng buộc của phần tử sẽ hiển thị.
5. Thực hiện theo hình minh họa bên dưới cho bước này. Nhấp vào tay cầm tròn ở bên phải của **TextView** để xóa ràng buộc ngang ràng buộc phần tử này với cạnh phải của bố cục. **TextView** sẽ di chuyển sang bên trái vì nó không còn bị ràng buộc với cạnh phải nữa. Để thêm lại ràng buộc ngang, nhấp vào cùng tay cầm đó và kéo một đường đến cạnh phải của bố cục.

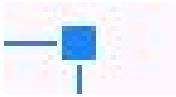


Trong bảng thiết kế hoặc bảng bản vẽ, các tay cầm sau xuất hiện trên phần tử TextView:

- Tay cầm ràng buộc: Để tạo một ràng buộc như trong hình minh họa động ở trên, nhấp vào tay cầm ràng buộc, được hiển thị dưới dạng một vòng tròn ở cạnh của phần tử. Sau đó, kéo tay cầm đến một tay cầm ràng buộc khác hoặc đến ranh giới của phần tử cha. Một đường gấp khúc sẽ thể hiện ràng buộc.



- Tay cầm thay đổi kích thước: Để thay đổi kích thước của phần tử, kéo các tay cầm thay đổi kích thước hình vuông. Khi kéo, tay cầm sẽ chuyển thành một góc xiên.

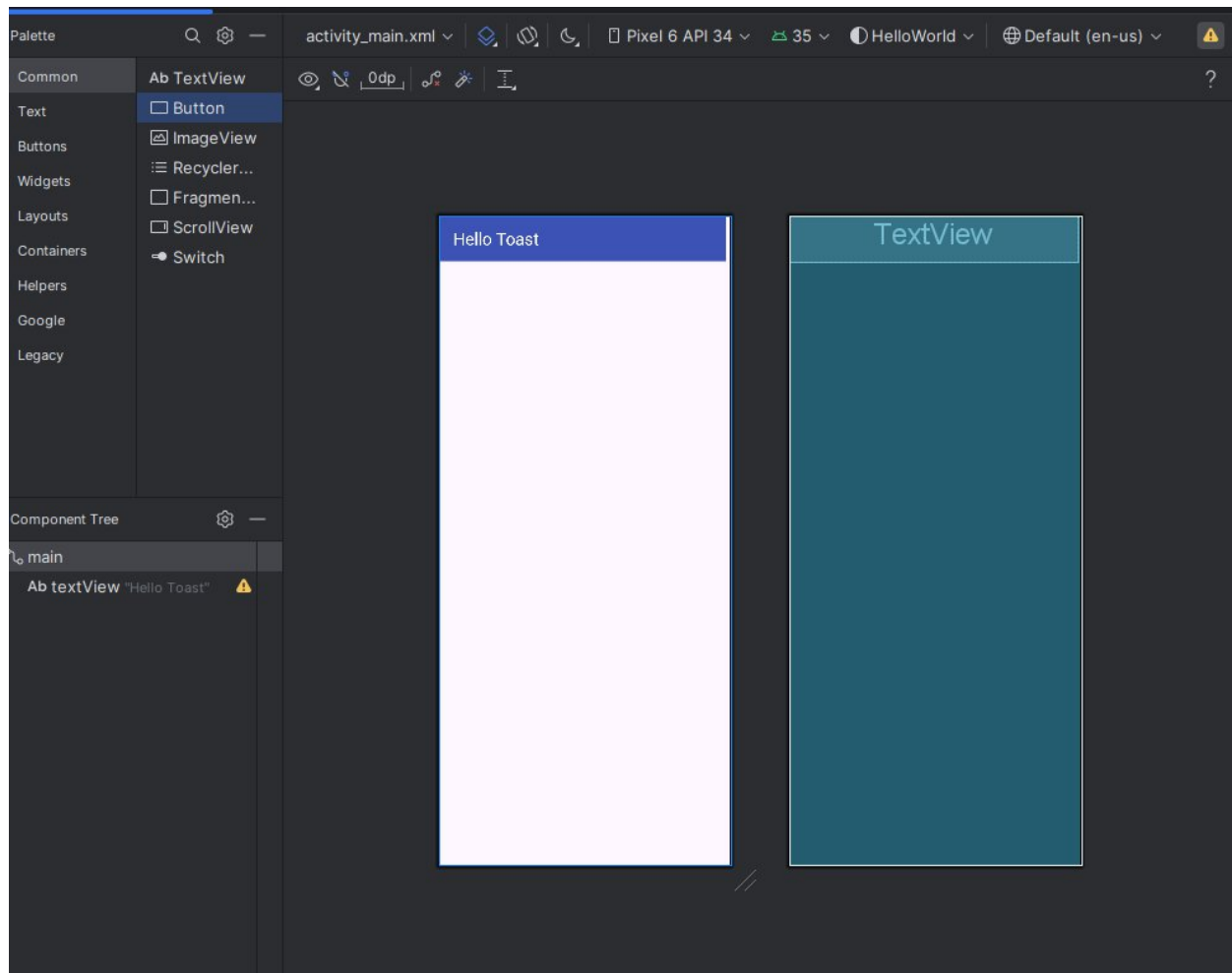


## 2.2 Thêm một Button vào bố cục

Khi được bật, công cụ **Autoconnect** sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử UI với bố cục cha. Sau khi bạn kéo phần tử vào bố cục, nó sẽ tạo ràng buộc dựa trên vị trí của phần tử.

Thực hiện các bước sau để thêm một **Button**:

1. Bắt đầu với một bố cục trống. Phần tử **TextView** không cần thiết, vì vậy khi nó vẫn đang được chọn, nhấn phím **Delete** hoặc chọn **Edit > Delete**. Bây giờ bạn có một bố cục hoàn toàn trống.
2. Kéo một **Button** từ bảng **Palette** vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** vào khu vực giữa phía trên của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo ràng buộc đến mép trên, bên trái và bên phải của bố cục như trong hình động bên dưới.



## 2.3 Add Button thứ 2 vào bố cục

1. Kéo một **Button** khác từ bảng **Palette** vào giữa bố cục như trong hình động bên dưới. **Autoconnect** có thể tự động tạo ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo chúng).

2. Kéo một ràng buộc dọc xuống mép dưới của bố cục (tham khảo hình bên dưới).



Bạn có thể xóa ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuột qua để hiển thị nút **Clear Constraints**. Nhấn vào nút này để xóa tất cả ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm ràng

buộc tương ứng. Để xóa tất cả ràng buộc trong toàn bộ bố cục, nhấn vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này hữu ích nếu bạn muốn thiết lập lại toàn bộ ràng buộc trong bố cục của mình.

## Bước 3 : Thay đổi thuộc tính của phần tử UI

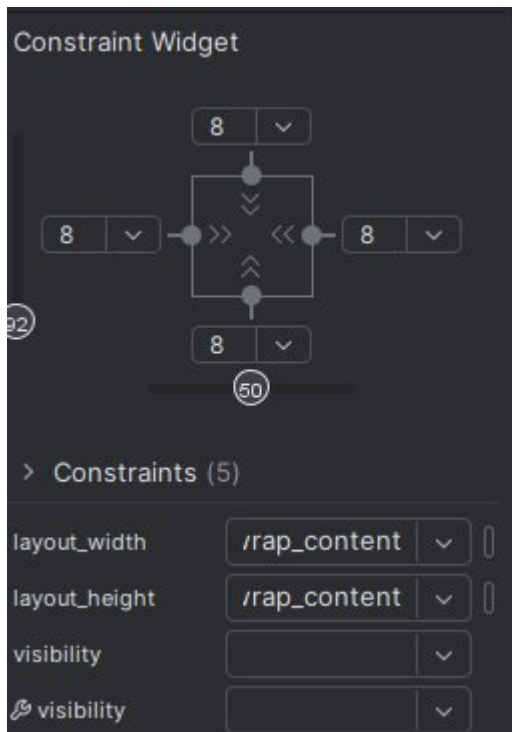
Ngăn Thuộc tính (**Attributes**) cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng (UI). Bạn có thể tìm thấy các thuộc tính chung cho tất cả các View trong tài liệu của lớp View.

Trong nhiệm vụ này, bạn sẽ nhập các giá trị mới và thay đổi các giá trị của các thuộc tính quan trọng của Button, những thuộc tính này cũng áp dụng cho hầu hết các loại View.

### 3.1 Thay đổi kích cỡ của Button

Trình chỉnh sửa bố cục cung cấp các tay nắm thay đổi kích thước ở bốn góc của một View, giúp bạn có thể nhanh chóng điều chỉnh kích thước của nó. Bạn có thể kéo các tay nắm ở mỗi góc của View để thay đổi kích thước, nhưng làm như vậy sẽ đặt cứng các kích thước chiều rộng và chiều cao. Tránh đặt kích thước cố định cho hầu hết các phần tử View, vì các kích thước cố định không thể thích ứng với nội dung và các kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn Thuộc tính (Attributes) ở phía bên phải của trình chỉnh sửa bố cục để chọn một chế độ kích thước không sử dụng kích thước cố định. Ngăn Thuộc tính bao gồm một bảng điều khiển kích thước hình vuông, được gọi là view inspector, ở phía trên. Các biểu tượng bên trong hình vuông biểu thị các cài đặt chiều cao và chiều rộng như sau:

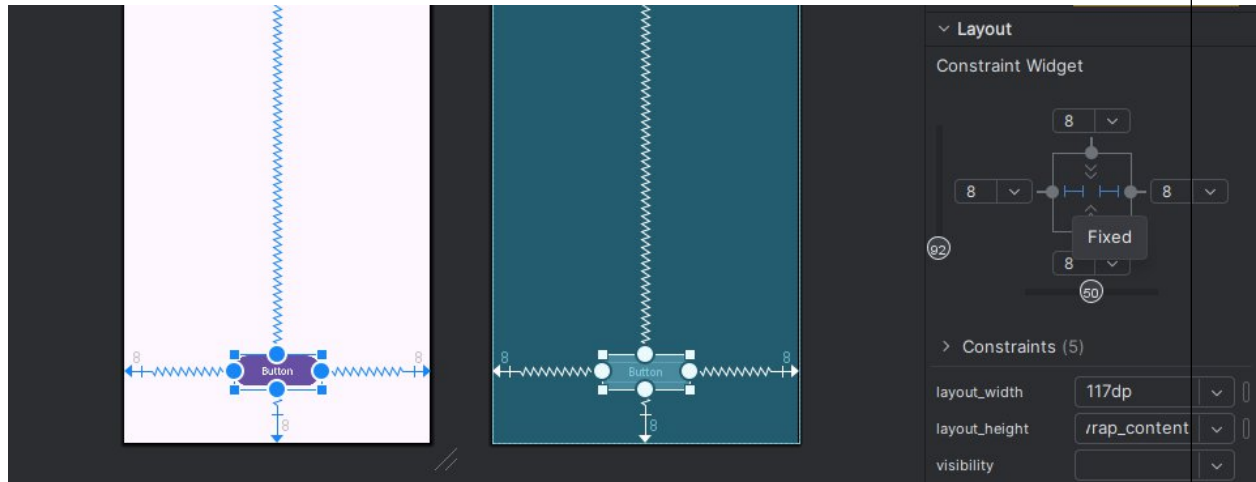


Trong hình trên:

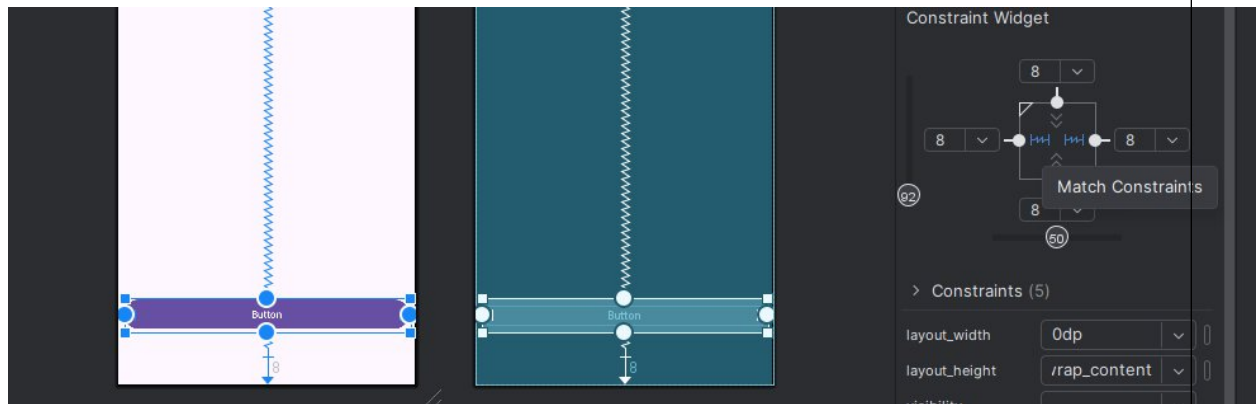
1. **Điều khiển chiều cao:** Điều khiển này xác định thuộc tính `layout_height` và xuất hiện dưới dạng hai đoạn ở phía trên và dưới của hình vuông. Các góc nghiêng cho thấy rằng điều khiển này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần thiết để phù hợp với nội dung của nó. Số "8" biểu thị một lề tiêu chuẩn được đặt là 8dp.
2. **Điều khiển chiều rộng:** Điều khiển này xác định thuộc tính `layout_width` và xuất hiện dưới dạng hai đoạn ở phía bên trái và bên phải của hình vuông. Các góc nghiêng cho thấy rằng điều khiển này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần thiết để phù hợp với nội dung của nó, với lề tối đa là 8dp.
3. **Nút đóng ngăn Thuộc tính:** Nhấp vào để đóng ngăn này.

Thực hiện các bước sau:

1. Chọn **nút (Button) trên cùng** trong ngăn **Component Tree**.
2. Nhấp vào **tab Attributes** ở phía bên phải của cửa sổ **layout editor**.
3. Nhấp **hai lần** vào điều khiển chiều rộng:
  - o Lần nhấp đầu tiên thay đổi thành **Fixed** (đường thẳng).



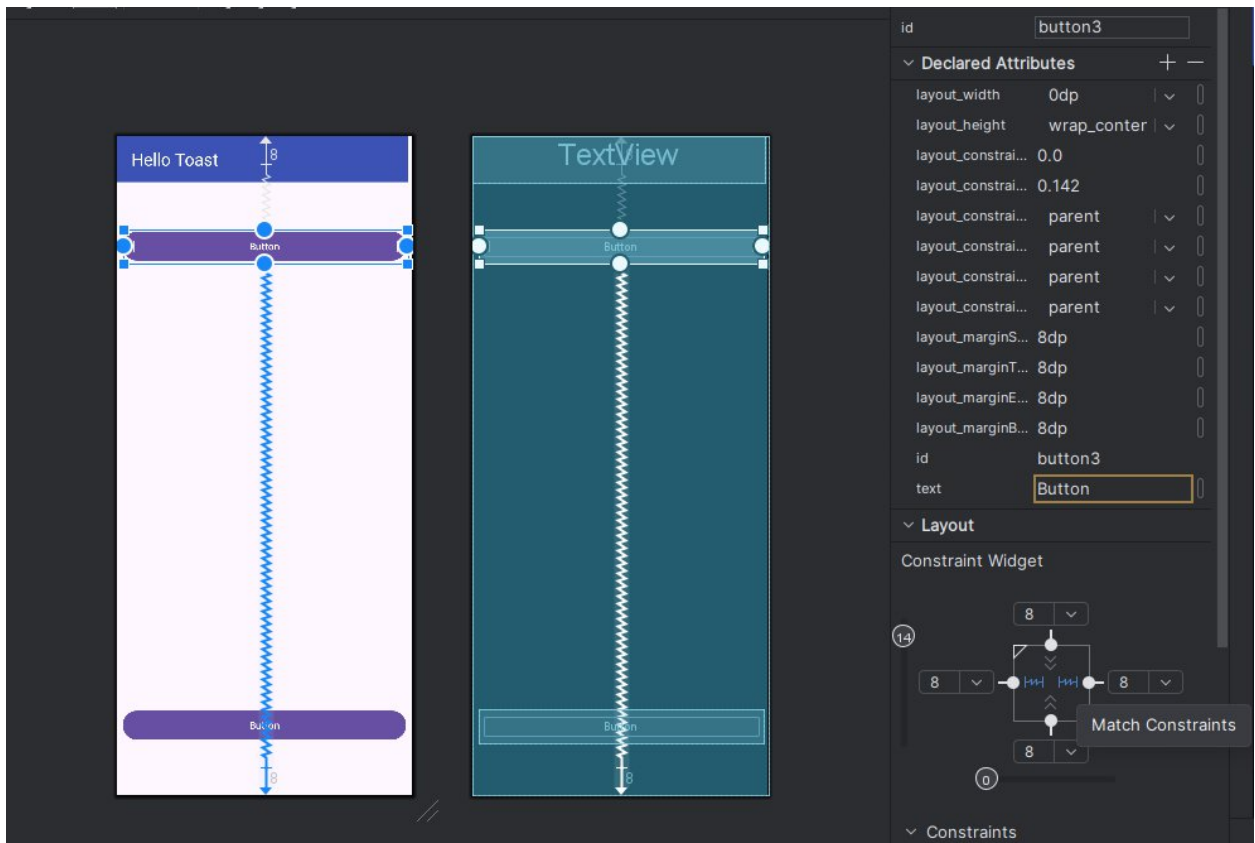
- Lần nhấp thứ hai thay đổi thành **Match Constraints** (lò xo cuộn), như trong hình động bên dưới.



Khi thay đổi điều khiển chiều rộng, thuộc tính `layout_width` trong **ngăn Attributes** hiển thị giá trị **match\_constraint**, và phần tử **Button** sẽ kéo giãn theo chiều ngang để lấp đầy khoảng trống giữa hai bên của layout.

4. Chọn **nút (Button) thứ hai**, và thực hiện thay đổi `layout_width` tương tự như bước trước đó, như trong hình minh họa bên dưới.





Như đã trình bày ở các bước trước, các thuộc tính `layout_width` và `layout_height` trong ngăn **Attributes** sẽ thay đổi khi bạn điều chỉnh các điều khiển chiều rộng và chiều cao trong **inspector**. Các thuộc tính này có thể nhận một trong ba giá trị sau trong **ConstraintLayout**:

- **match\_constraint**: Mở rộng phần tử **View** để lấp đầy **parent** theo chiều rộng hoặc chiều cao—tối đa đến **margin** nếu được đặt. Trong trường hợp này, **parent** chính là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
- **wrap\_content**: Thu nhỏ kích thước của phần tử **View** sao cho nó vừa đủ để bao bọc nội dung bên trong. Nếu không có nội dung, **View** sẽ trở nên **vô hình**.
- **Fixed size (kích thước cố định)**: Để đặt kích thước cố định mà vẫn có thể thích ứng với kích thước màn hình của thiết bị, hãy sử dụng đơn vị **density-independent pixels (dp)**. Ví dụ: 16dp có nghĩa là **16 pixel độc lập** với mật độ màn hình.

**Mẹo:** Nếu bạn thay đổi thuộc tính `layout_width` bằng menu bật lên của nó, giá trị của `layout_width` sẽ được đặt thành **0** vì không có kích thước cố định nào được thiết lập. Thiết lập này tương đương với **`match_constraint`**—phần tử có thể mở rộng tối đa để đáp ứng các ràng buộc và khoảng lề đã đặt.

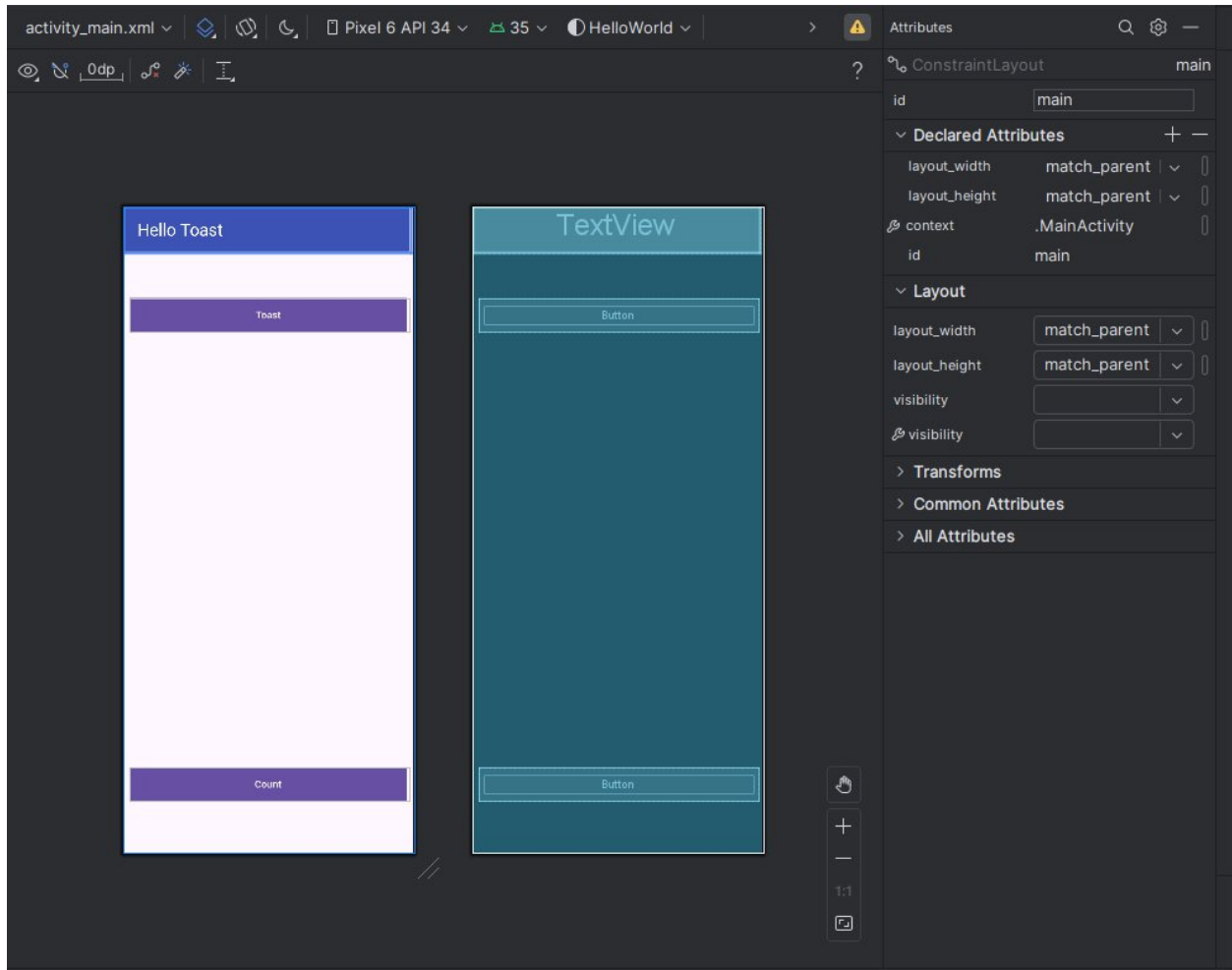
### 3.2 Thay đổi thuộc tính của Button

Để xác định duy nhất từng View trong một Activity layout, mỗi View hoặc lớp con của View (chẳng hạn như Button) cần có một ID duy nhất. Ngoài ra, các phần tử Button cần có văn bản để sử dụng được. Các phần tử View cũng có thể có nền là màu sắc hoặc hình ảnh.

**Bảng thuộc tính** cung cấp quyền truy cập vào tất cả các thuộc tính có thể gán cho một phần tử View. Bạn có thể nhập giá trị cho từng thuộc tính, chẳng hạn như `android:id`, `background`, `textColor` và `text`.

Hình động sau đây minh họa cách thực hiện các bước này:

1. Sau khi chọn Button đầu tiên, chỉnh sửa trường ID ở đầu Bảng thuộc tính thành **`button_toast`** cho thuộc tính `android:id`, dùng để xác định phần tử trong bố cục.
2. Đặt thuộc tính `background` thành **`@color/colorPrimary`**. (Khi bạn nhập **`@c`**, các tùy chọn sẽ xuất hiện để dễ dàng lựa chọn.)
3. Đặt thuộc tính `textColor` thành **`@android:color/white`**.
4. Chỉnh sửa thuộc tính `text` thành **`Toast`**.



- Thực hiện các thay đổi thuộc tính tương tự cho **Button** thứ hai, sử dụng `button_count` làm **ID**, đặt thuộc tính text thành **Count**, và sử dụng cùng màu cho background và text như các bước trước.

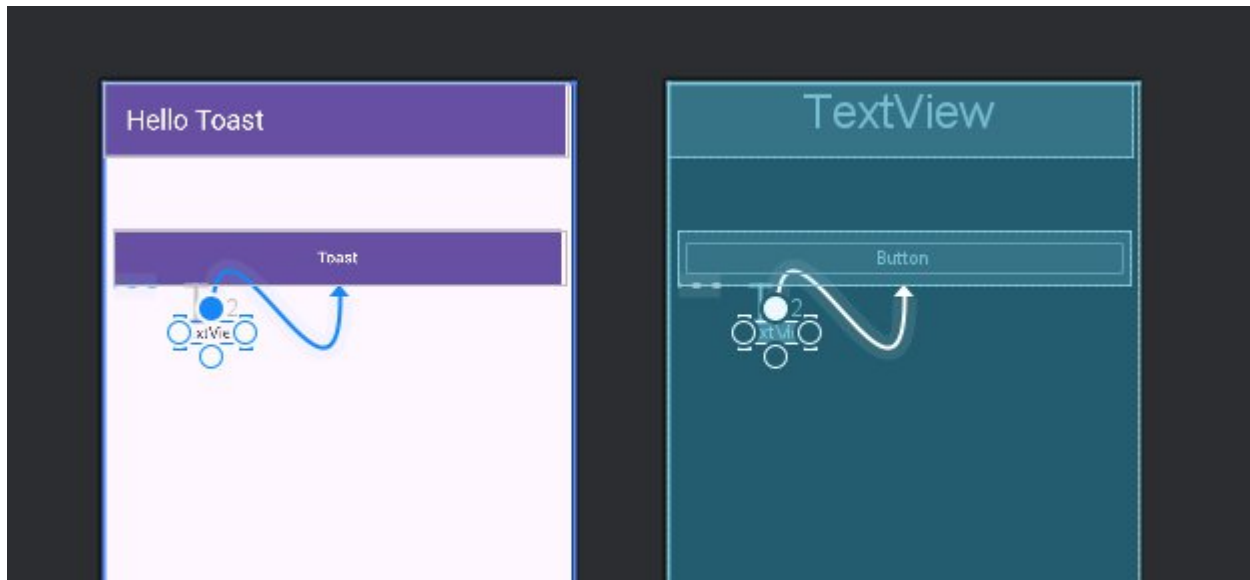
## Bước 4: Thêm **TextEdit** và đặt thuộc tính cho nó

Một trong những lợi ích của **ConstraintLayout** là khả năng căn chỉnh hoặc ràng buộc các phần tử so với các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một **TextView** vào giữa bố cục và ràng buộc nó theo chiều ngang với các lề và theo chiều dọc với hai **Button**. Sau đó, bạn sẽ thay đổi các thuộc tính cho **TextView** trong **Attributes** pane.

### 4.1 Thêm **TextView** và ràng buộc

- Như hình động bên dưới, kéo một **TextView** từ ngăn **Palette** vào phần trên của bố cục, và kéo một ràng buộc từ đỉnh của **TextView** đến tay cầm ở phía dưới của **Button Toast**. Việc này sẽ ràng buộc **TextView** nằm bên dưới **Button**.

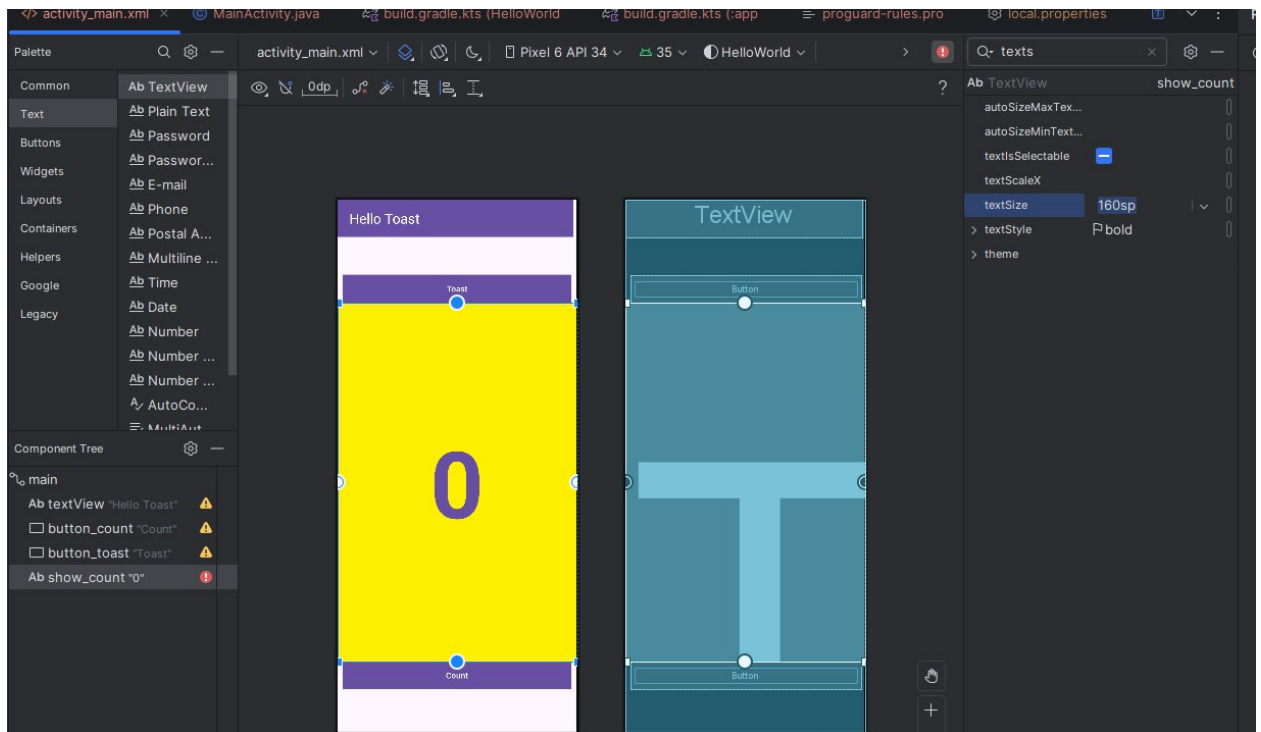
2. Như hình động bên dưới, kéo một ràng buộc từ đáy của **TextView** đến tay cầm ở phía trên của **Button Count**, và từ hai bên của **TextView** đến hai bên của bố cục. Việc này sẽ ràng buộc **TextView** nằm giữa bố cục, giữa hai **Button**.



## 4.2 Thiết lập thuộc tính cho TextView

Với **TextView** được chọn, mở ngăn **Attributes** nếu nó chưa được mở. Thiết lập các thuộc tính cho **TextView** như trong hình động bên dưới. Các thuộc tính mà bạn chưa gặp sẽ được giải thích sau hình

1. Đặt **ID** thành `show_count`.
2. Đặt **text** thành 0.
3. Đặt **textSize** thành 160sp.
4. Đặt **textStyle** thành **B** (đậm) và **textAlignment** thành `ALIGNCENTER` (căn giữa đoạn văn bản).
5. Thay đổi điều khiển kích thước ngang và dọc (**layout\_width** và **layout\_height**) thành `match_constraint`.
6. Đặt **textColor** thành `@color/colorPrimary`.
7. Cuộn xuống ngăn **Attributes**, nhấp vào **View all attributes**, cuộn xuống trang thứ hai đến **background**, và nhập `#FFF00` để đặt màu vàng nhạt.
8. Cuộn xuống **gravity**, mở rộng mục này và chọn `center_ver` (căn giữa theo chiều dọc).



- **textSize**: Kích thước văn bản của **TextView**. Trong bài học này, kích thước được đặt thành 160sp. **sp** (scale-independent pixel) là đơn vị tỷ lệ theo mật độ màn hình và cài đặt kích thước phông chữ của người dùng. Khi chỉ định kích thước phông chữ, nên sử dụng đơn vị **sp** để đảm bảo văn bản điều chỉnh phù hợp với cả mật độ màn hình và tùy chọn của người dùng.
- **textStyle** và **textAlignment**: **textStyle** được đặt thành **B** (đậm) và **textAlignment** được đặt thành **ALIGNCENTER** (căn giữa đoạn văn bản).
- **gravity**: Thuộc tính **gravity** xác định cách một **View** được căn chỉnh trong **View** hoặc **ViewGroup** cha. Trong bước này, **TextView** được căn giữa theo chiều dọc trong **ConstraintLayout**.

Bạn có thể nhận thấy rằng thuộc tính **background** nằm ở trang đầu của ngăn **Attributes** đối với **Button**, nhưng lại nằm ở trang thứ hai đối với **TextView**. Ngăn **Attributes** thay đổi tùy theo loại **View**: Các thuộc tính phổ biến nhất của loại **View** sẽ xuất hiện trên trang đầu tiên, trong khi các thuộc tính khác được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của ngăn **Attributes**, hãy nhấp vào biểu tượng trong thanh công cụ ở đầu ngăn.

## Bước 5: Chỉnh bố cục trong XML

Bố cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử giao diện người dùng trong Component Tree. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hiển


thị bên dưới. Cùng một cảnh báo xuất hiện cho cả ba phần tử: các chuỗi được mã hóa cứng (hardcoded strings) nên sử dụng tài nguyên.



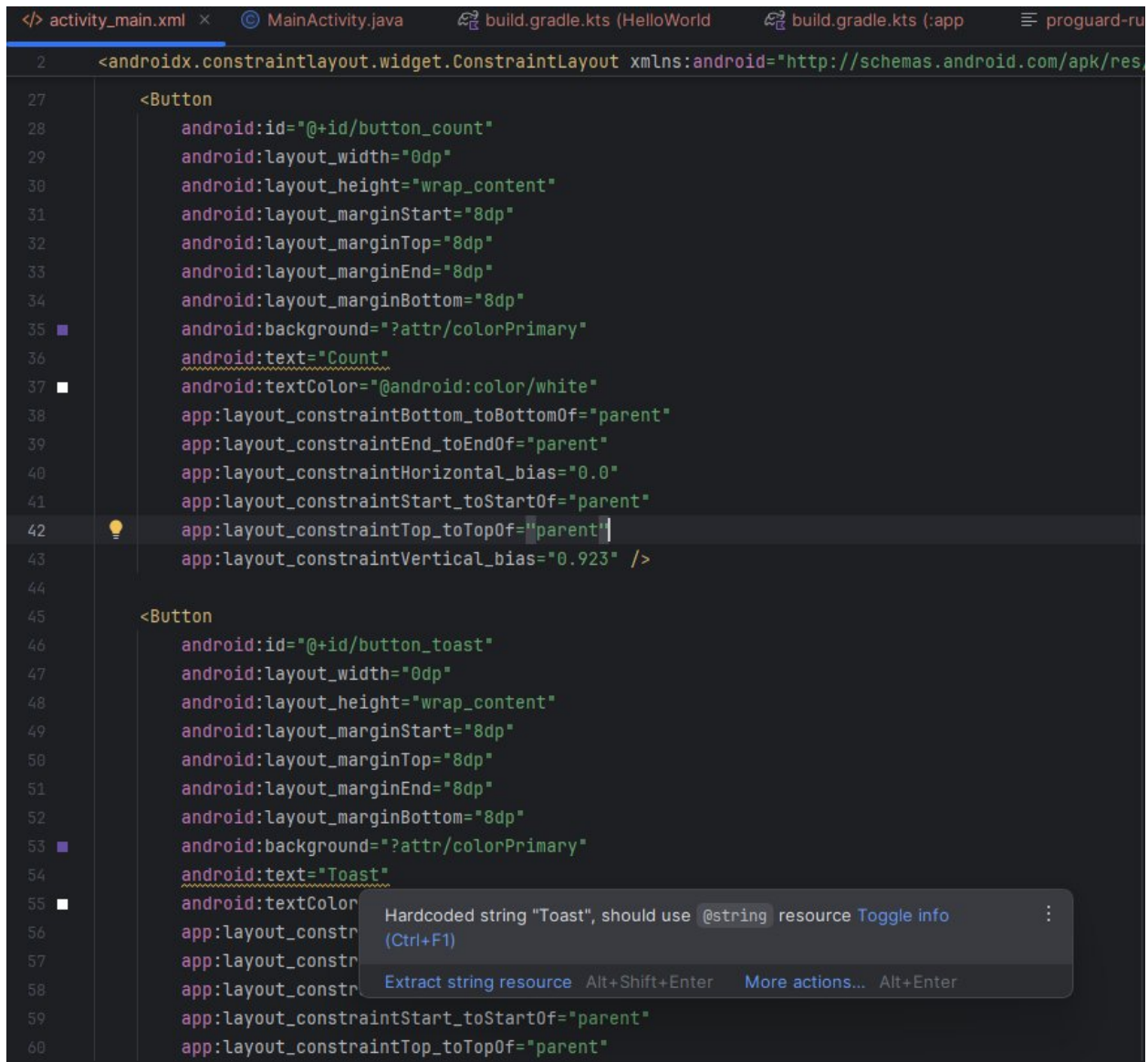
Cách dễ nhất để khắc phục sự cố bố cục là chỉnh sửa bố cục trong XML. Mặc dù trình chỉnh sửa bố cục là một công cụ mạnh mẽ, nhưng một số thay đổi sẽ dễ thực hiện hơn khi chỉnh sửa trực tiếp trong mã nguồn XML.

## 5.1 Mở mã XML cho bố cục

Đối với nhiệm vụ này, mở tệp **activity\_main.xml** nếu nó chưa được mở và nhấp

vào tab **Text**  ở dưới cùng của trình chỉnh sửa bố cục.

Trình chỉnh sửa XML xuất hiện, thay thế các ngăn thiết kế và bản vẽ. Như bạn có thể thấy trong hình dưới đây, hiển thị một phần mã XML của bố cục, các cảnh báo được đánh dấu—các chuỗi mã cứng **"Toast"** và **"Count"**. (Chuỗi mã cứng **"0"** cũng được đánh dấu nhưng không hiển thị trong hình.) Di chuột qua chuỗi mã cứng **"Toast"** để xem thông báo cảnh báo.



```
<?xml version="1.0" encoding="utf-8" android:namespace="http://schemas.android.com/apk/res/android"
    2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res
    27      <Button
    28          android:id="@+id/button_count"
    29          android:layout_width="0dp"
    30          android:layout_height="wrap_content"
    31          android:layout_marginStart="8dp"
    32          android:layout_marginTop="8dp"
    33          android:layout_marginEnd="8dp"
    34          android:layout_marginBottom="8dp"
    35          android:background="?attr/colorPrimary"
    36          android:text="Count"
    37          android:textColor="@android:color/white"
    38          app:layout_constraintBottom_toBottomOf="parent"
    39          app:layout_constraintEnd_toEndOf="parent"
    40          app:layout_constraintHorizontal_bias="0.0"
    41          app:layout_constraintStart_toStartOf="parent"
    42          app:layout_constraintTop_toTopOf="parent"
    43          app:layout_constraintVertical_bias="0.923" />
    44
    45      <Button
    46          android:id="@+id/button_toast"
    47          android:layout_width="0dp"
    48          android:layout_height="wrap_content"
    49          android:layout_marginStart="8dp"
    50          android:layout_marginTop="8dp"
    51          android:layout_marginEnd="8dp"
    52          android:layout_marginBottom="8dp"
    53          android:background="?attr/colorPrimary"
    54          android:text="Toast"
    55          android:textColor="@android:color/white"
    56          app:layout_constraintBottom_toBottomOf="parent"
    57          app:layout_constraintEnd_toEndOf="parent"
    58          app:layout_constraintHorizontal_bias="0.0"
    59          app:layout_constraintStart_toStartOf="parent"
    60          app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 5.2 Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng chuỗi, một phương pháp hay nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Việc đặt các chuỗi trong một tệp riêng biệt giúp dễ dàng quản lý chúng hơn, đặc biệt nếu bạn sử dụng những chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và địa phương hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhấp một lần vào từ "Toast" (cảnh báo được tô sáng đầu tiên).
2. Nhấn **Alt-Enter** trên Windows hoặc **Option-Enter** trên macOS, sau đó chọn **Extract string resource** từ menu bật lên.
3. Nhập **button\_label\_toast** cho **Resource name**.



4. Nhấn **OK**. Một tài nguyên chuỗi được tạo trong tệp **res/values/strings.xml**, và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: `@string/button_label_toast`.
5. Trích xuất các chuỗi còn lại:
  - **button\_label\_count** cho "Count"
  - **count\_initial\_value** cho "0"
6. Trong **Project > Android**, mở rộng **values** trong **res**, sau đó nhấp đúp vào **strings.xml** để xem các tài nguyên chuỗi trong tệp **strings.xml**.

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_toast">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    ⚡ <string name="count_initial_value">0</string>
</resources>
```

7. Bạn cần một chuỗi khác để sử dụng trong nhiệm vụ tiếp theo nhằm hiển thị một thông báo. Thêm vào tệp **strings.xml** một tài nguyên chuỗi mới có tên **toast\_message** với nội dung "Hello Toast!":

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_toast">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
    ⚡ <string name="toast_message">Hello Toast!</string>
</resources>
```

**Mẹo:** Các tài nguyên chuỗi bao gồm tên ứng dụng, tên này sẽ xuất hiện trên thanh ứng dụng ở đầu màn hình nếu bạn khởi tạo dự án ứng dụng bằng mẫu **Empty Template**. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên **app\_name**.

## Bước 6: Thêm trình xử lý **onClick** cho các nút

Trong nhiệm vụ này, bạn thêm một phương thức Java cho mỗi nút trong **MainActivity**, phương thức này sẽ được thực thi khi người dùng nhấn vào nút.

### 6.1 Thêm thuộc tính **onClick** và trình xử lý cho từng Button



Trình xử lý sự kiện nhấp chuột là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường **onClick** của **Attributes pane** ở tab **Design**. Bạn cũng có thể chỉ định tên của phương thức xử lý trong **trình chỉnh sửa XML** bằng cách thêm thuộc tính `android:onClick` vào nút **Button**. Bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý, và trình chỉnh sửa XML cung cấp một cách tự động để tạo các phương thức đó.

1. Khi trình chỉnh sửa XML đang mở (tab **Text**), tìm **Button** có thuộc tính `android:id` được đặt thành `button_toast`.
2. Thêm thuộc tính `android:onClick` vào cuối phần tử `button_toast`, sau thuộc tính cuối cùng và trước dấu kết thúc `</>`.
3. Nhấp vào biểu tượng **bóng đèn đỏ** xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity**, rồi nhấp **OK**.
  - Nếu không thấy biểu tượng **bóng đèn đỏ**, nhấp vào tên phương thức ("`showToast`"), nhấn `Alt+Enter` (hoặc `Option+Enter` trên Mac), chọn **Create 'showToast(view)' in MainActivity**, rồi nhấp **OK**.
  - Hành động này sẽ tạo một phương thức xử lý sự kiện (`showToast()`) trong **MainActivity**.
4. Lặp lại các bước trên cho **Button** có `android:id="button_count"`:
  - Thêm thuộc tính `android:onClick`.
  - Tạo phương thức xử lý sự kiện.

Mã XML cho các phần tử giao diện người dùng trong `ConstraintLayout` bây giờ trông như thế này:

```

<Button
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="?attr/colorPrimary"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.923"
    android:onClick="showCount"/>

```

```

<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="?attr/colorPrimary"
    android:text="@string/button_label_toast"
    android:textColor="@android:color/white"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.142"
    android:onClick="showToast" />

```

5. Nếu **MainActivity.java** chưa mở, hãy mở **Project > Android**, mở rộng thư mục java, sau đó mở rộng com.example.android.hellotoast, và nhấp đúp vào **MainActivity.java** để mở trình chỉnh sửa mã.

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        Log.d("MainActivity", "Hello World");
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }

    1 usage
    public void showToast(View view) {
    }

    1 usage
    public void showCount(View view) {
    }
}

```

## 6.2 Chỉnh sửa trình xử lý Button Toast

Bây giờ bạn sẽ chỉnh sửa phương thức `showToast()`—trình xử lý sự kiện nhấn nút Toast trong `MainActivity`—để hiển thị một thông báo. Toast cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ, chỉ chiếm không gian cần thiết cho nội dung thông báo. Hoạt động hiện tại vẫn hiển thị và có thể tương tác. Toast hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn—bạn có thể thêm một thông báo Toast để hiển thị kết quả khi nhấn một Button hoặc thực hiện một hành động nào đó.

Thực hiện các bước sau để chỉnh sửa trình xử lý sự kiện của Button Toast:

1. Xác định phương thức `showToast()` vừa được tạo.

```

1 usage
public void showToast(View view) {
}

```

2. Tạo một đối tượng Toast bằng cách gọi phương thức `makeText()` của lớp Toast.

3. Cung cấp ngữ cảnh (context) của Activity. Vì Toast hiển thị trên giao diện của Activity, hệ thống cần thông tin về Activity hiện tại. Khi đang ở trong Activity cần sử dụng context, bạn có thể dùng this làm shortcut.
4. Cung cấp thông báo cần hiển thị, chẳng hạn như một chuỗi tài nguyên (string resource) toast\_message mà bạn đã tạo ở bước trước. Chuỗi tài nguyên toast\_message được xác định bằng R.string.toast\_message.
5. Cung cấp thời gian hiển thị. Ví dụ, Toast.LENGTH\_SHORT hiển thị Toast trong thời gian ngắn.
  - Toast.LENGTH\_LONG: Hiển thị khoảng 3,5 giây.
  - Toast.LENGTH\_SHORT: Hiển thị khoảng 2 giây.
6. Hiển thị Toast bằng cách gọi show().

```
1 usage
public void showToast(View view) {
    Toast toast = Toast.makeText(context: this, R.string.toast_message, Toast.LENGTH_SHORT);
    toast.show();
}
```

Sau khi hoàn thành, hãy chạy ứng dụng và kiểm tra xem thông báo Toast có xuất hiện khi nhấn Button Toast hay không.

07:27

100%

Hello Toast

Toast

0



Hello Toast!



## 6.3 Chỉnh sửa trình xử lý sự kiện của Button Count

Bây giờ bạn sẽ chỉnh sửa phương thức `countUp()`—trình xử lý sự kiện của nút Count trong `MainActivity`—để hiển thị số đếm hiện tại sau khi nhấn Count. Mỗi lần nhấn sẽ tăng số đếm lên một đơn vị.

Mã cho trình xử lý phải:

- Theo dõi số đếm khi nó thay đổi.
- Gửi số đếm đã cập nhật đến `TextView` để hiển thị.

Thực hiện các bước sau để chỉnh sửa trình xử lý nút Count:

1. Xác định phương thức `countUp()` vừa được tạo.
2. Để theo dõi số đếm, bạn cần một biến thành viên riêng tư. Mỗi lần nhấn nút Count sẽ tăng giá trị của biến này. Nhập đoạn mã sau, đoạn mã này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn đỏ:

```
no usages
public void countUp(View view) {
    mCount++;
}
```

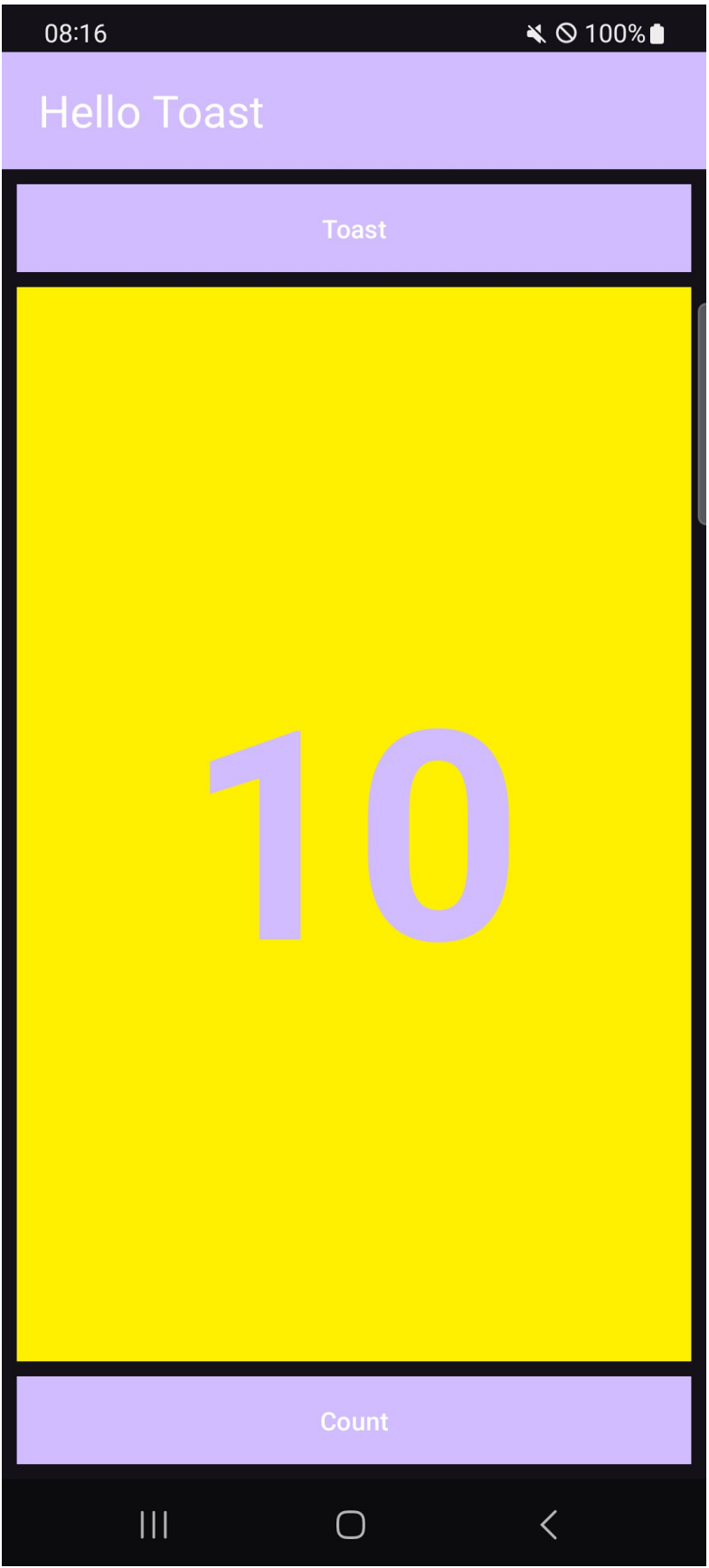
Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy chọn biểu thức `mCount++`. Biểu tượng bóng đèn cuối cùng sẽ xuất hiện.

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Điều này tạo một biến thành viên riêng tư ở đầu `MainActivity`, và Android Studio giả định rằng bạn muốn nó là một số nguyên (`int`).
4. Thay đổi câu lệnh biến thành viên riêng tư để khởi tạo biến với giá trị bằng 0.
5. Cùng với biến trên, bạn cũng cần một biến thành viên riêng tư để tham chiếu đến `TextView` có ID `show_count`, biến này sẽ được thêm vào trình xử lý nhấp chuột. Đặt tên biến là `mShowCount`.
6. Bây giờ bạn đã có `mShowCount`, bạn có thể lấy tham chiếu đến `TextView` bằng cách sử dụng ID bạn đã đặt trong tệp bố cục. Để lấy tham chiếu này chỉ một lần, hãy xác định nó trong phương thức `onCreate()`. Như bạn sẽ học trong một bài học khác, phương thức `onCreate()` được sử dụng để nạp bố cục, có nghĩa là thiết lập giao diện màn hình thành bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các thành phần giao diện khác trong bố cục, chẳng hạn như `TextView`.
7. Thêm câu lệnh `findViewById` vào cuối phương thức:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
    setContentView(R.layout.activity_main);  
    mShowCount = (TextView) findViewById(R.id.show_count);A
```

Một View, giống như một chuỗi, là một tài nguyên có thể có ID. Lệnh `findViewById` nhận ID của một View làm tham số và trả về View. Vì phương thức này trả về một View, bạn phải ép kiểu kết quả về kiểu `TextView`.

8. Sau khi đã gán `TextView` vào biến `mShowCount`, bạn có thể sử dụng biến này để đặt văn bản trong `TextView` thành giá trị của biến `mCount`. Thêm đoạn mã sau vào phương thức `countUp()`.
9. Chạy ứng dụng để kiểm tra xem số đếm có tăng khi bạn nhấn nút Count hay không.





# Tổng kết

View, ViewGroup và Layouts:

- Tất cả các phần tử giao diện người dùng (UI elements) đều là lớp con của lớp View và do đó kế thừa nhiều thuộc tính của lớp View cha.
- Các phần tử View có thể được nhóm lại bên trong một ViewGroup, hoạt động như một vùng chứa. Mỗi quan hệ này là cha - con, trong đó cha là ViewGroup, còn con là View hoặc một ViewGroup khác.
- Phương thức onCreate() được sử dụng để nạp bố cục (inflate layout), nghĩa là thiết lập nội dung màn hình thành bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong bố cục.
- Một View, giống như một chuỗi, là một tài nguyên có thể có id. Lệnh findViewById nhận ID của một View làm tham số và trả về đối tượng View tương ứng.

Sử dụng trình chỉnh sửa layout:

- Nhấp vào tab **Design** để thao tác với các phần tử và bố cục, hoặc tab **Text** để chỉnh sửa mã XML của bố cục.
- Trong tab **Design**, **Palettes** pane hiển thị các phần tử giao diện có thể sử dụng trong bố cục của ứng dụng. **Component tree** pane hiển thị cấu trúc phân cấp của các phần tử giao diện.
- Design pane và Blueprint pane của trình chỉnh sửa bố cục hiển thị các phần tử giao diện có trong bố cục.
- **Attributes tab** hiển thị bảng **Attributes** để thiết lập thuộc tính cho một phần tử giao diện.
- Constraint handle: Nhấp vào chấm tròn ở mỗi cạnh của một phần tử và kéo đến một chấm tròn khác hoặc ranh giới của ViewGroup cha để tạo ràng buộc (constraint). Ràng buộc này được biểu diễn bằng đường gấp khúc.
- Resizing handle: Kéo các ô vuông để thay đổi kích thước phần tử. Khi kéo, tay cầm thay đổi thành một góc xiên.
- Autoconnect tool (công cụ kết nối tự động) tự động tạo hai hoặc nhiều ràng buộc cho một phần tử giao diện với bố cục cha dựa trên vị trí của nó.
- Xóa ràng buộc: Chọn phần tử, di chuột qua để hiển thị nút Clear Constraints, nhấp vào để xóa tất cả ràng buộc. Để xóa một ràng buộc cụ thể, nhấp vào chấm tròn của ràng buộc đó.
- **Attributes pane** chứa tất cả thuộc tính XML có thể áp dụng cho một phần tử giao diện. Nó cũng có **View Inspector**, một bảng điều chỉnh kích thước

vuông nhỏ ở đầu. Các biểu tượng bên trong biểu thị các cài đặt về chiều rộng và chiều cao.

Cài đặt chiều rộng và chiều cao của bố cục:

Các thuộc tính `layout_width` và `layout_height` thay đổi theo điều chỉnh kích thước trong View Inspector. Với `ConstraintLayout`, có ba giá trị có thể sử dụng:

- `match_constraint`: Mở rộng phần tử để lấp đầy vùng chứa theo chiều rộng hoặc chiều cao, giới hạn bởi phần margin (nếu có).
- `wrap_content`: Thu nhỏ kích thước phần tử sao cho vừa với nội dung của nó. Nếu không có nội dung, phần tử trở nên vô hình.
- Giá trị số cố định (dp - density-independent pixels): Định kích thước cố định, điều chỉnh theo kích thước màn hình thiết bị.

Trích xuất chuỗi tài nguyên (string resources):

1. Nhấp vào chuỗi cần trích xuất, nhấn **Alt + Enter** (**Option + Enter** trên Mac) và chọn **Extract string resources** từ menu bật lên.
2. Đặt tên cho tài nguyên chuỗi (**Resource name**).
3. Nhấp **OK**. Một tài nguyên chuỗi được tạo trong tệp `res/values/strings.xml`, và chuỗi trong mã sẽ được thay thế bằng một **tham chiếu tài nguyên**: `@string/button_label_toast`.

Xử lý sự kiện nhấp:

- **Click handler** là một phương thức được gọi khi người dùng **nhấp hoặc chạm** vào một phần tử giao diện.
- Để chỉ định một click handler cho một phần tử giao diện (như Button), **Trong tab Design**, nhập tên phương thức vào trường **onClick** trong bảng **Attributes** hoặc trong XML, thêm thuộc tính `android:onClick` vào phần tử (Button).
- **Tạo phương thức click handler** trong MainActivity với tham số View. Ví dụ: `public void showToast(View view) {...}`.
- Bạn có thể tìm thấy tất cả thuộc tính của Button trong tài liệu lớp **Button**, và thuộc tính của TextView trong tài liệu lớp **TextView**.

Hiển thị thông báo Toast:

Toast là một cách để hiển thị một thông báo nhỏ trong popup, chỉ chiếm không gian cần thiết cho nội dung. Các bước tạo Toast:

1. Gọi phương thức `makeText()` từ lớp `Toast`.
2. Cung cấp **context** của ứng dụng (Activity) và **chuỗi thông báo** (có thể là tài nguyên chuỗi).
3. Xác định **thời gian hiển thị**, `Toast.LENGTH_SHORT`: Hiển thị trong thời gian ngắn. `Toast.LENGTH_LONG`: Hiển thị trong thời gian dài.
4. Gọi `show()` để hiển thị `Toast`.

### 1.3) Trình chỉnh sửa bố cục

## Mở đầu

Như bạn đã học trong **Phần 1.2: Giao diện người dùng tương tác đầu tiên**, bạn có thể xây dựng giao diện người dùng (UI) bằng **ConstraintLayout** trong trình chỉnh sửa bố cục (layout editor), bố cục này sắp xếp các phần tử UI bằng cách sử dụng các ràng buộc (constraint) kết nối với các phần tử khác và với các cạnh của bố cục. **ConstraintLayout** được thiết kế để giúp bạn dễ dàng kéo các phần tử UI vào trình chỉnh sửa bố cục.

**ConstraintLayout** là một **ViewGroup**, một loại View đặc biệt có thể chứa các đối tượng View khác (gọi là **children** hoặc **child views**). Bài thực hành này giới thiệu thêm nhiều tính năng của **ConstraintLayout** và trình chỉnh sửa bố cục.

Bài thực hành này cũng giới thiệu hai lớp con khác của **ViewGroup**:

- **LinearLayout**: Một nhóm sắp xếp các phần tử View con bên trong nó theo chiều ngang hoặc chiều dọc.
- **RelativeLayout**: Một nhóm các phần tử View con, trong đó mỗi phần tử View được định vị và căn chỉnh tương đối với các phần tử khác trong **ViewGroup**. Vị trí của các phần tử View con được mô tả dựa trên mối quan hệ với nhau hoặc với **ViewGroup** cha.

## Những gì bạn nên biết

Bạn nên biết:

- Tạo ứng dụng Hello World bằng Android Studio.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị thật.
- Tạo bố cục đơn giản cho ứng dụng bằng **ConstraintLayout**.
- Trích xuất và sử dụng tài nguyên chuỗi (string resources).

## Những gì bạn sẽ học

- Cách tạo một biến thể bố cục cho chế độ ngang (landscape).
- Cách tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng ràng buộc đường cơ sở (baseline constraint) để căn chỉnh các phần tử giao diện người dùng với văn bản.
- Cách sử dụng các nút gói (pack) và căn chỉnh (align) để sắp xếp các phần tử trong bố cục.
- Cách định vị các View trong LinearLayout.
- Cách định vị các View trong RelativeLayout.

## **Những gì bạn sẽ làm**

- Tạo một biến thể bố cục cho chế độ hiển thị ngang.
- Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Chỉnh sửa bố cục để thêm ràng buộc (constraints) cho các phần tử giao diện người dùng.
- Sử dụng ràng buộc đường cơ sở (baseline constraints) trong ConstraintLayout để căn chỉnh các phần tử với văn bản.
- Sử dụng các nút gói (pack) và căn chỉnh (align) trong ConstraintLayout để sắp xếp các phần tử.
- Thay đổi bố cục để sử dụng LinearLayout.
- Định vị các phần tử trong LinearLayout.
- Thay đổi bố cục để sử dụng RelativeLayout.
- Sắp xếp lại các View trong bố cục chính sao cho liên kết tương đối với nhau.

## **Tổng quan ứng dụng**

Ứng dụng Hello Toast trong bài học trước sử dụng ConstraintLayout để sắp xếp các phần tử giao diện người dùng trong bố cục Activity, như hình minh họa bên dưới.

08:16

100%

Hello Toast

Toast

10

Count



Để thực hành nhiều hơn với `ConstraintLayout`, bạn sẽ tạo một biến thể của bố cục này cho chế độ ngang như trong hình minh họa bên dưới.

Bạn cũng sẽ học cách sử dụng ràng buộc đường cơ sở (baseline constraints) và một số tính năng căn chỉnh của `ConstraintLayout` bằng cách tạo một biến thể bố cục khác cho màn hình máy tính bảng.

Ngoài ra, bạn sẽ tìm hiểu về các lớp con khác của `ViewGroup`, chẳng hạn như `LinearLayout` và `RelativeLayout`, đồng thời thay đổi bố cục của ứng dụng Hello Toast để sử dụng chúng.

## Bước 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng Hello Toast để phù hợp với chế độ hiển thị ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể của bố cục cho điện thoại theo chiều ngang (còn gọi là *landscape*) và chiều dọc (còn gọi là *portrait*), cũng như cho các màn hình lớn hơn như máy tính bảng.

Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trong hai thanh công cụ trên cùng của trình chỉnh sửa bố cục. Thanh công cụ trên cùng cho phép bạn cấu hình giao diện của bản xem trước bố cục trong trình chỉnh sửa.



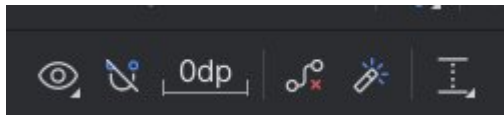
Trong hình minh họa trên:

- Chọn Bề mặt Thiết kế:** Chọn *Design* để hiển thị bản xem trước có màu của bố cục hoặc *Blueprint* để chỉ hiển thị đường viền của từng phần tử giao diện người dùng. Để xem cả hai gần cạnh nhau, chọn *Design + Blueprint*.
- Hướng trong Trình chỉnh sửa:** Chọn *Portrait* hoặc *Landscape* để hiển thị bản xem trước theo chiều dọc hoặc ngang. Điều này hữu ích để xem trước bố cục mà không cần chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo các bố cục thay thế, chọn *Create Landscape Variation* hoặc các biến thể khác.
- Thiết bị trong Trình chỉnh sửa:** Chọn loại thiết bị (*điện thoại/máy tính bảng, Android TV hoặc Android Wear*).
- Phiên bản API trong Trình chỉnh sửa:** Chọn phiên bản Android để hiển thị bản xem trước.

5. **Chủ đề trong Trình chỉnh sửa:** Chọn một chủ đề (*chẳng hạn như AppTheme*) để áp dụng cho bản xem trước.

6. **Ngôn ngữ trong Trình chỉnh sửa:** Chọn ngôn ngữ và khu vực hiển thị cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có sẵn trong tài nguyên chuỗi. Bạn cũng có thể chọn *Preview as Right To Left* để xem bố cục như thể đã chọn một ngôn ngữ RTL.

Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các phần tử UI trong *ConstraintLayout* và thu phóng hoặc di chuyển bản xem trước.



Trong hình trên:

1. **Show:** Chọn *Show Constraints* và *Show Margins* để hiển thị hoặc ẩn chúng trong bản xem trước.
2. **Autoconnect:** Bật hoặc tắt *Autoconnect*. Khi bật, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như *Button*) đến bất kỳ vị trí nào trong bố cục để tự động tạo ràng buộc với bố cục cha.
3. **Clear All Constraints:** Xóa tất cả các ràng buộc trong toàn bộ bố cục.
4. **Infer Constraints:** Tạo ràng buộc bằng cách suy luận.
5. **Default Margins:** Đặt lề mặc định.
6. **Pack:** Gom nhóm hoặc mở rộng các phần tử đã chọn.
7. **Align:** Căn chỉnh các phần tử đã chọn.
8. **Guidelines:** Thêm đường hướng dẫn dọc hoặc ngang.
9. **Zoom/pan controls:** Thu phóng hoặc di chuyển.

**Mẹo:** Để tìm hiểu thêm về cách sử dụng *layout editor*, hãy xem **Build a UI with Layout Editor**. Để tìm hiểu thêm về cách xây dựng bố cục với *ConstraintLayout*, hãy xem **Build a Responsive UI with ConstraintLayout**.

## 1.1 Xem trước bố cục trong chế độ ngang

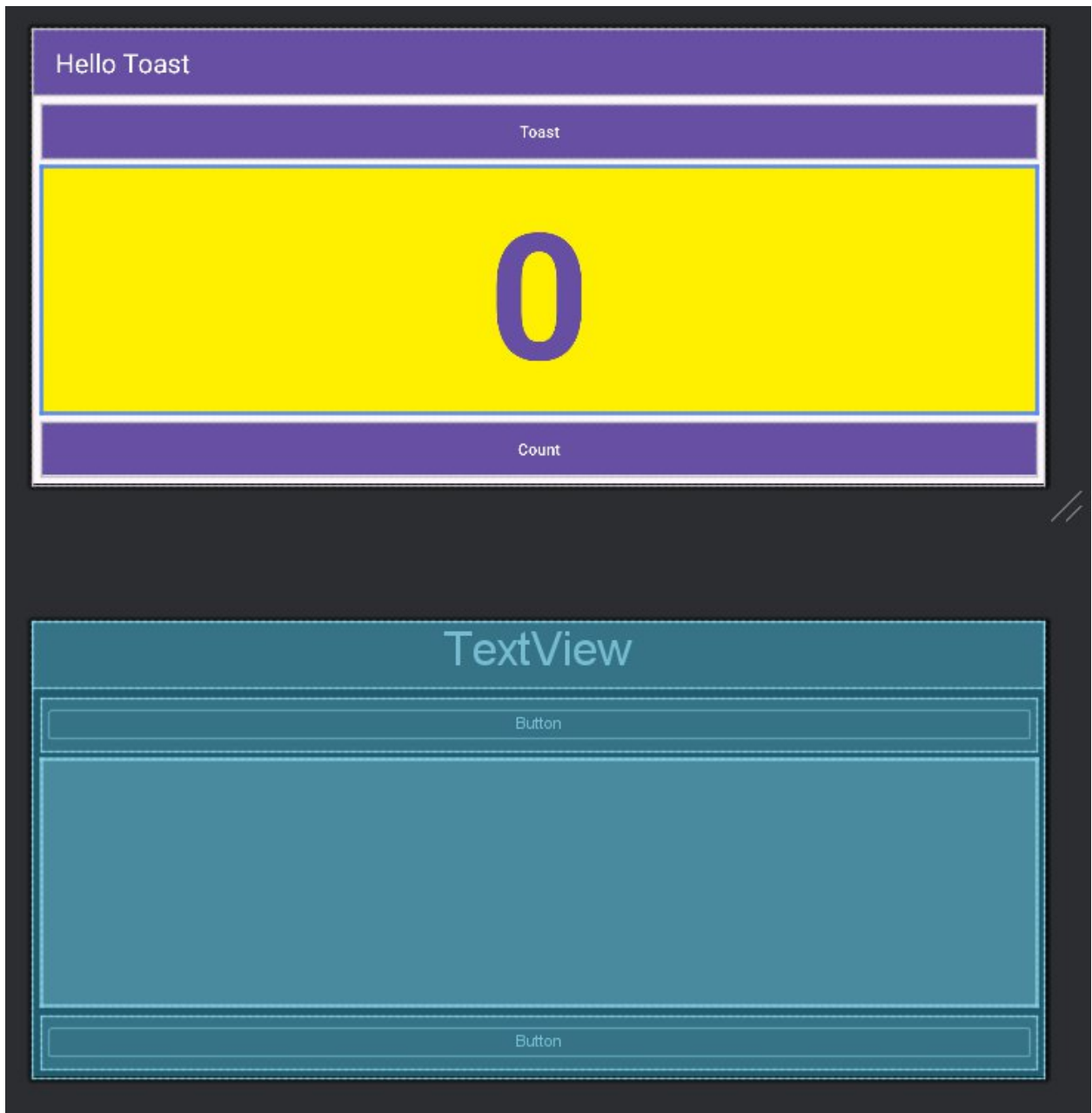
Để xem trước bố cục của ứng dụng Hello Toast trong chế độ ngang, hãy làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ bài học trước. **Lưu ý:** Nếu bạn đã tải xuống mã nguồn giải pháp của HelloToast, bạn cần xóa các bố cục landscape và extra-large đã hoàn thành mà bạn sẽ tạo trong nhiệm vụ này. Chuyển từ

**Project > Android** sang **Project > Project Files** trong bảng Project. Mở rộng đường dẫn: app > src/main > res, chọn cả hai thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**. Sau đó, chuyển bảng Project trở lại **Project > Android**.

2. Mở tệp bố cục activity\_main.xml. Nhấp vào tab **Design** nếu nó chưa được chọn.
3. Nhấp vào nút **Orientation in Editor** trên thanh công cụ phía trên.
4. Chọn **Switch to Landscape** trong menu thả xuống. Bố cục sẽ hiển thị trong chế độ ngang như hình dưới đây. Để quay lại chế độ dọc, chọn **Switch to Portrait**



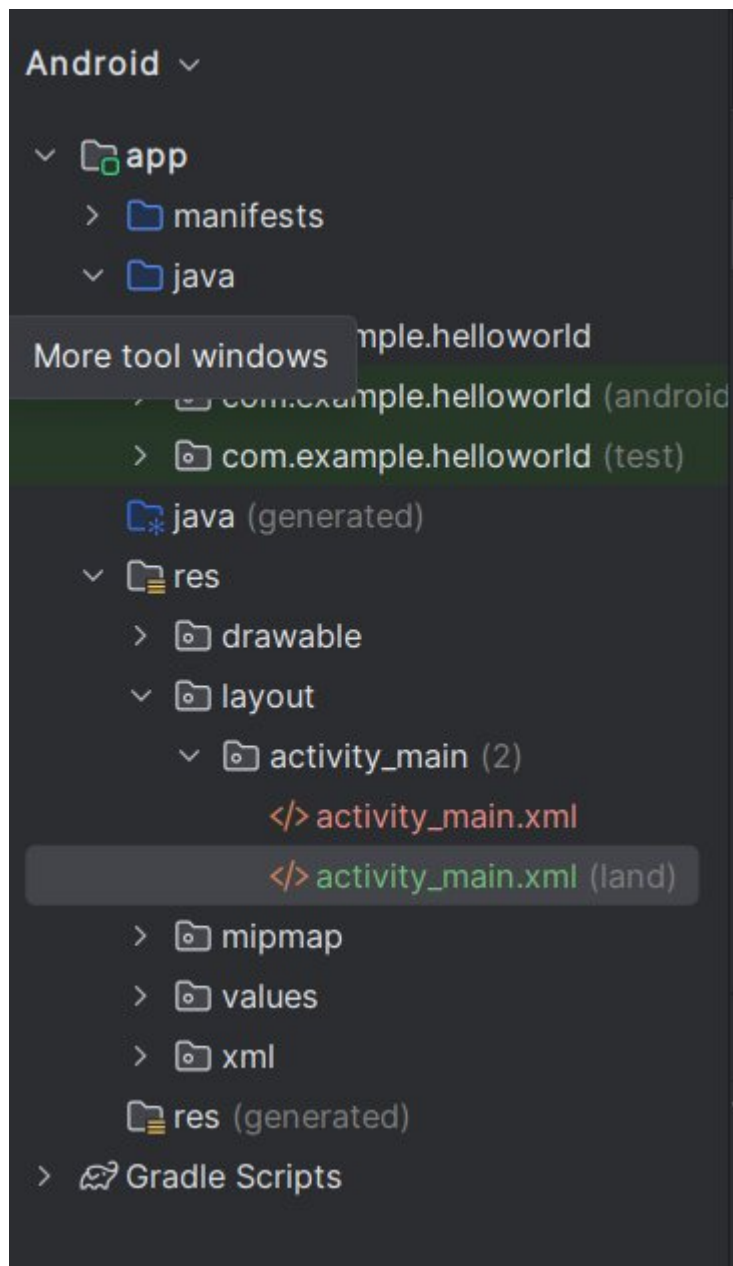


## 1.2 Tạo một biến thể bố cục cho chế độ ngang

Sự khác biệt về mặt hiển thị giữa chế độ dọc và chế độ ngang trong bố cục này là chữ số (0) trong phần tử `TextView show_count` nằm quá thấp trong chế độ ngang—quá gần với nút `Count`. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử `TextView` có thể hiển thị quá lớn hoặc không căn giữa vì kích thước văn bản được cố định ở 160sp.

Để khắc phục vấn đề này trong chế độ ngang mà không ảnh hưởng đến chế độ dọc, bạn có thể tạo một biến thể của bố cục ứng dụng Hello Toast dành riêng cho chế độ ngang. Hãy làm theo các bước sau:

1. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
2. Chọn **Create Landscape Variation**. Một cửa sổ chỉnh sửa mới sẽ mở ra với tab `land/activity_main.xml`, hiển thị bố cục dành cho chế độ ngang (horizontal). Bạn có thể chỉnh sửa bố cục này dành riêng cho chế độ ngang mà không ảnh hưởng đến bố cục gốc ở chế độ dọc (vertical).
3. Trong ngăn **Project > Android**, mở thư mục `res > layout`, bạn sẽ thấy Android Studio đã tự động tạo một biến thể có tên `activity_main.xml (land)`.



### 1.3 Xem trước bố cục trên các thiết bị khác nhau

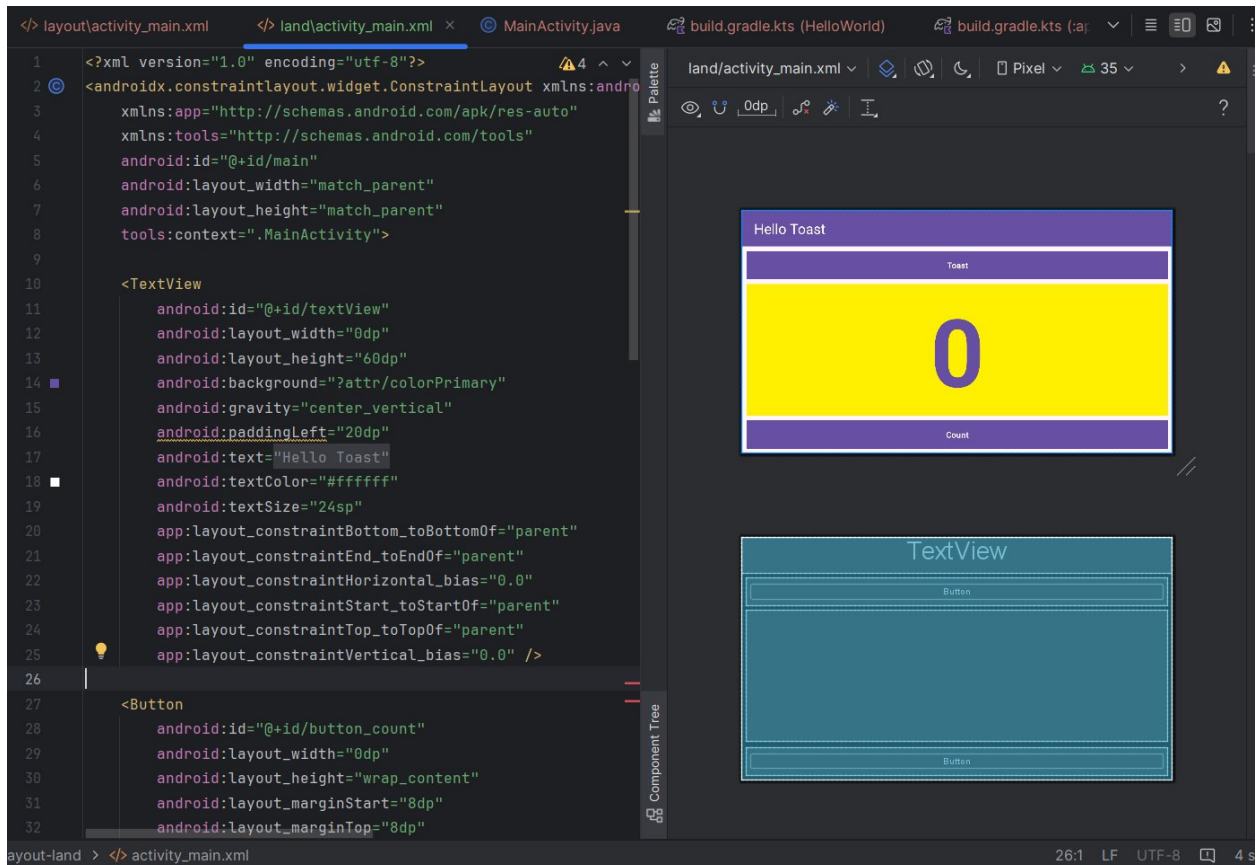
Bạn có thể xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thực hoặc trình giả lập. Thực hiện các bước sau:

1. Tab **land/activity\_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục; nếu không, hãy nhấp đúp vào tệp **activity\_main.xml (land)** trong thư mục **layout**.
2. Nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng.

3. Chọn một thiết bị khác trong menu thả xuống. Ví dụ, chọn **Nexus 4**, **Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong các bản xem trước. Những khác biệt này là do kích thước văn bản cố định trong **TextView**.

## 1.4 Thay đổi bố cục cho chế độ ngang

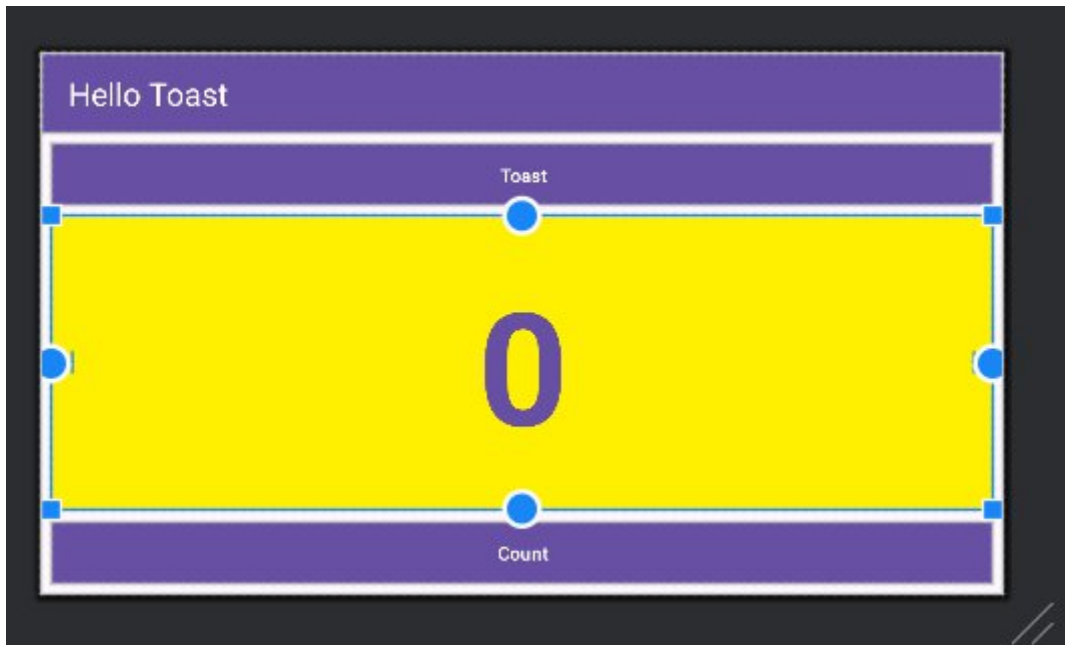
Bạn có thể sử dụng ngăn **Attributes** trong tab **Design** để thiết lập hoặc thay đổi thuộc tính, nhưng đôi khi chỉnh sửa trực tiếp mã XML trong tab **Text** sẽ nhanh hơn. Tab **Text** hiển thị mã XML và cung cấp tab **Preview** ở bên phải cửa sổ để xem trước bố cục, như minh họa trong hình dưới đây.



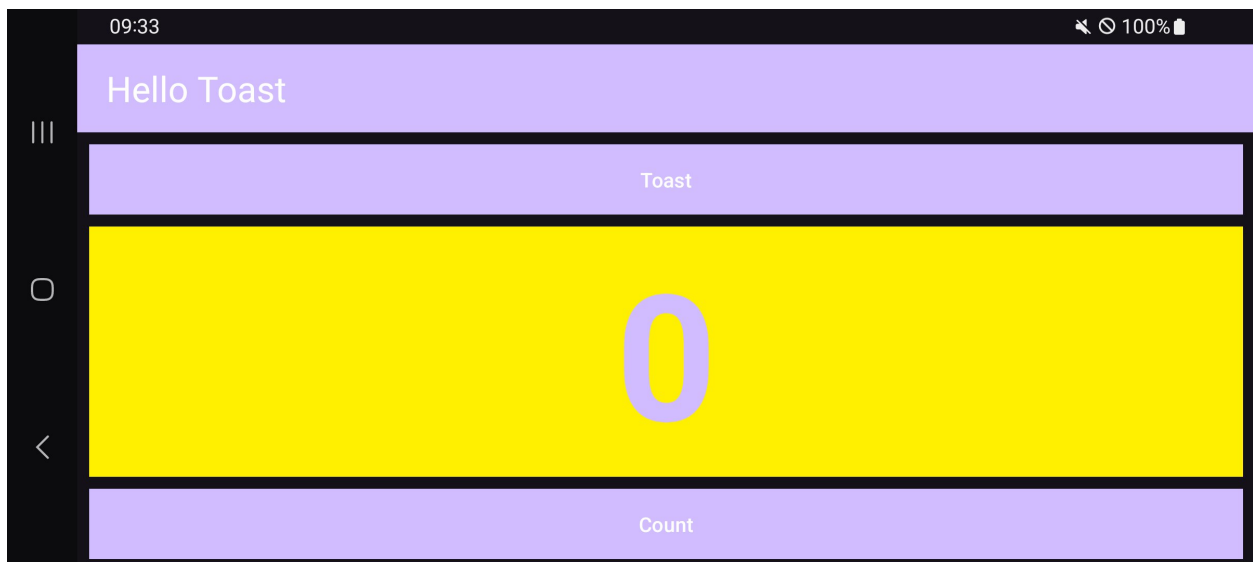
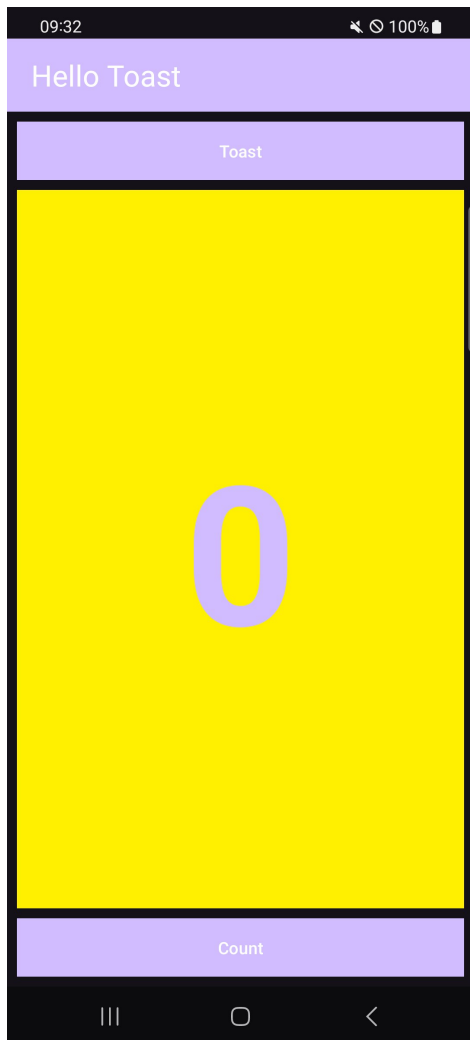
Để thay đổi bố cục, hãy làm theo các bước sau:

1. Tab **land/activity\_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục; nếu không, hãy nhấp đúp vào tệp **activity\_main.xml (land)** trong thư mục **layout**.
2. Nhấp vào tab **Text** và tab **Preview** (nếu chưa được chọn).
3. Tìm phần tử **TextView** trong mã XML.

4. Thay đổi thuộc tính `android:textSize="160sp"` thành `android:textSize="120sp"`. Bản xem trước bố cục sẽ hiển thị kết quả:

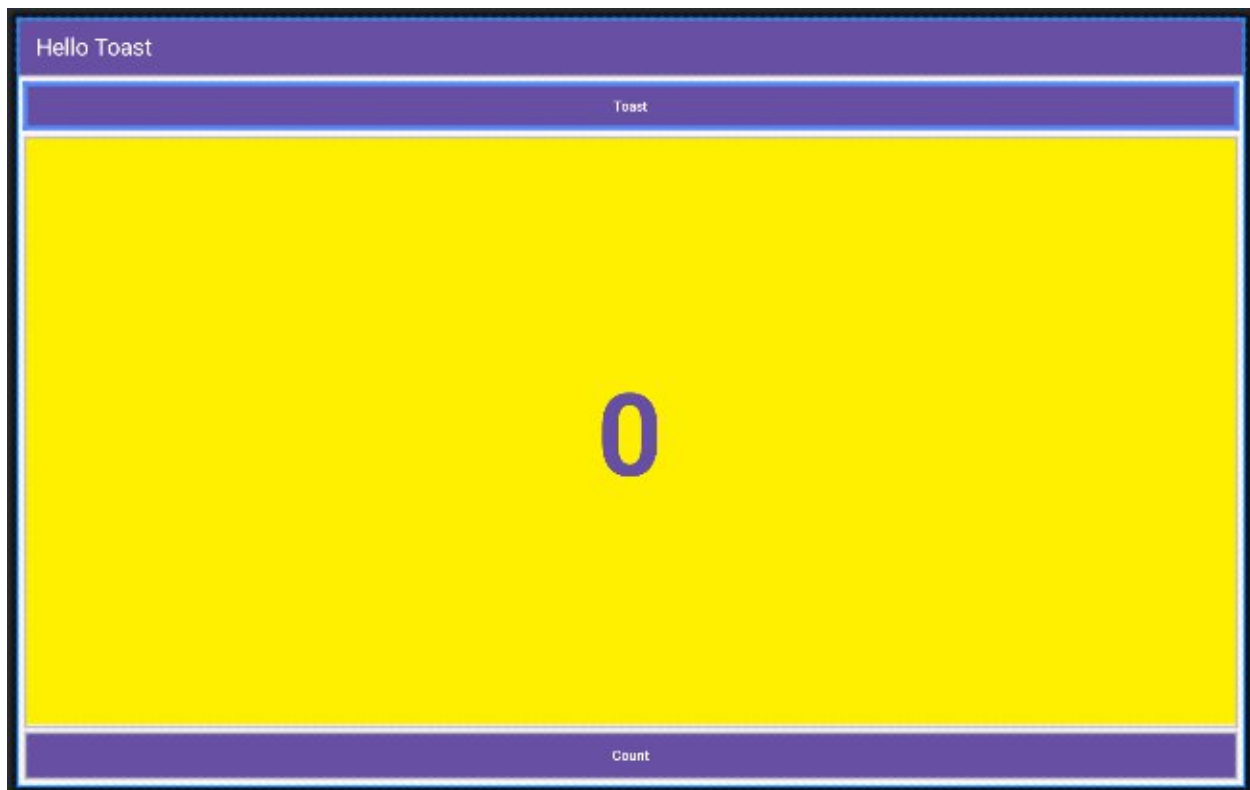


5. Chọn các thiết bị khác nhau trong menu thả xuống **Device in Editor** để xem bố cục hiển thị như thế nào trên các thiết bị khác nhau ở chế độ ngang. Trong ngăn chỉnh sửa, tab **land/activity\_main.xml** hiển thị bố cục cho chế độ ngang. Tab **activity\_main.xml** hiển thị bố cục chưa thay đổi cho chế độ dọc. Bạn có thể chuyển qua lại bằng cách nhấp vào các tab.
6. Chạy ứng dụng trên trình giả lập hoặc thiết bị thật, sau đó chuyển đổi giữa chế độ dọc và ngang để xem cả hai bố cục.



## 1.5 Tạo một biến thể bố cục cho máy tính bảng

Như bạn đã học trước đó, bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng. Nếu bạn chọn một thiết bị như **Nexus 10** (một máy tính bảng) từ menu, bạn sẽ thấy rằng bố cục không phù hợp với màn hình máy tính bảng—văn bản trên mỗi nút **Button** quá nhỏ và cách sắp xếp các nút ở trên cùng và dưới cùng không lý tưởng cho một màn hình lớn.



Để khắc phục điều này cho máy tính bảng trong khi vẫn giữ nguyên bố cục cho điện thoại ở cả chế độ dọc và ngang, bạn có thể tạo một biến thể bố cục hoàn toàn khác dành riêng cho máy tính bảng. Hãy làm theo các bước sau:

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị các bảng thiết kế và sơ đồ.
2. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
3. Chọn **Create layout x-large Variation**.

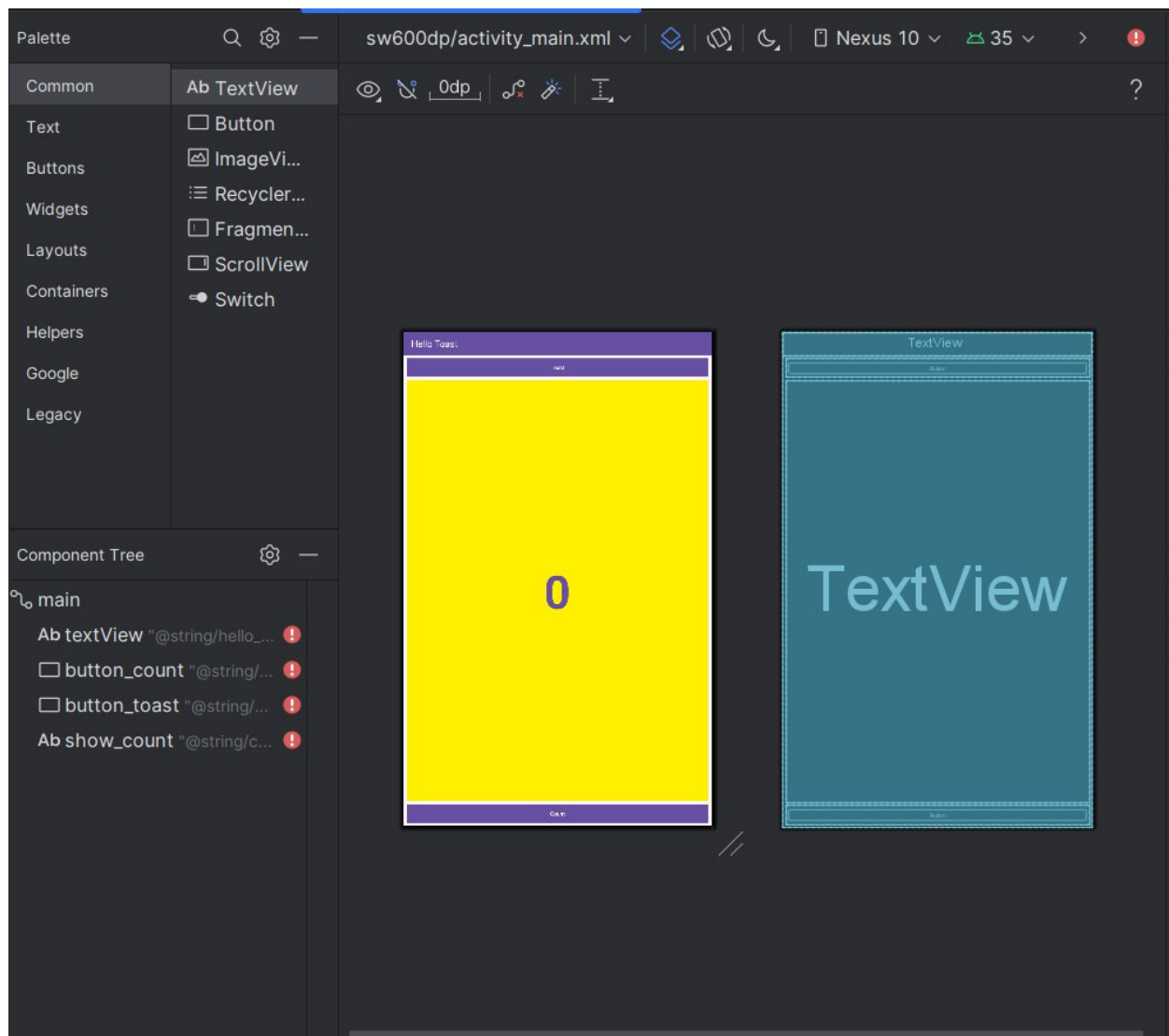
Một cửa sổ chỉnh sửa mới mở ra với tab **xlarge/activity\_main.xml**, hiển thị bố cục cho thiết bị có kích thước màn hình máy tính bảng. Trình chỉnh sửa cũng tự động chọn một thiết bị máy tính bảng, chẳng hạn như Nexus 9 hoặc Nexus 10, để hiển thị bản xem trước. Bạn có thể chỉnh sửa bố cục này dành riêng cho máy tính bảng mà không ảnh hưởng đến các bố cục khác.

## 1.6 Thay đổi biến thể bố cục cho máy tính bảng

Bạn có thể sử dụng ngăn Attributes trong tab **Design** để thay đổi thuộc tính cho bố cục này.

1. Tắt công cụ Autoconnect trên thanh công cụ. Đảm bảo rằng công cụ này đã bị vô hiệu hóa.
2. Xóa tất cả ràng buộc trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ. Khi đã xóa ràng buộc, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trên bố cục.
3. Thay đổi kích thước phần tử , trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở bốn góc của mỗi phần tử. Trong **Component Tree**, chọn TextView có ID show\_count. Để dễ dàng kéo các nút (Button), hãy kéo một góc của TextView để thay đổi kích thước của nó, như minh họa trong hình động bên dưới.

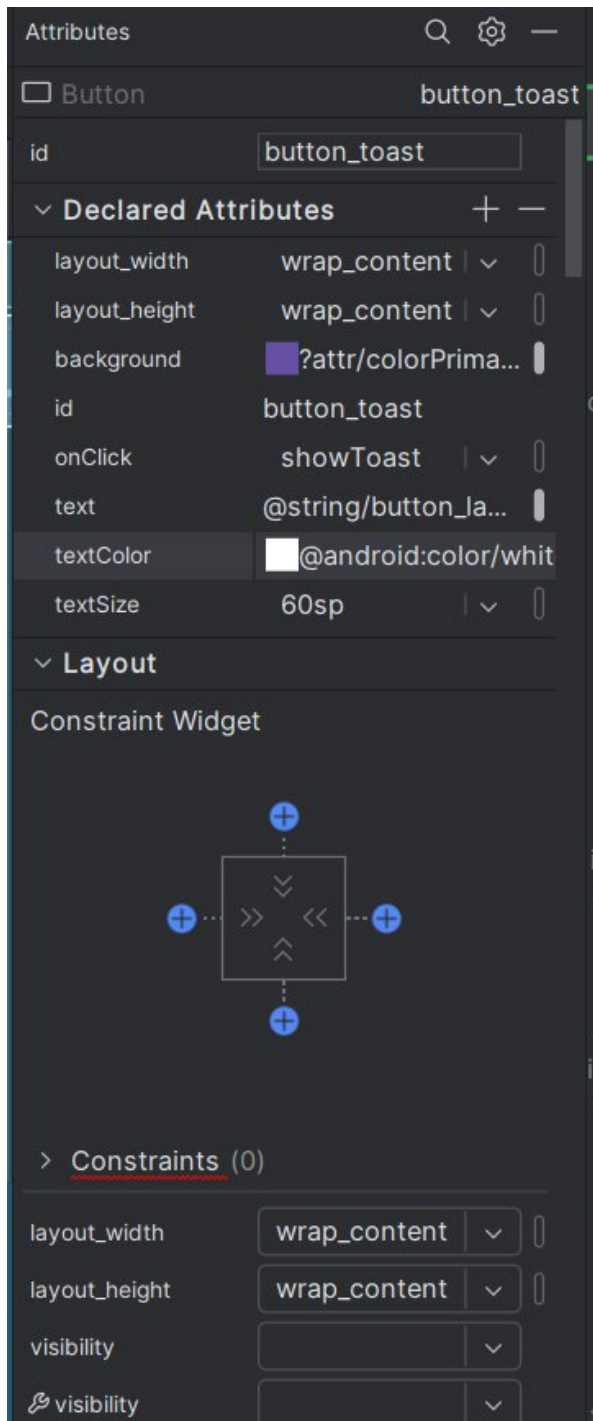




Việc thay đổi kích thước một phần tử sẽ gán cứng (hardcode) các giá trị chiều rộng và chiều cao. Tránh gán cứng kích thước cho hầu hết các phần tử, vì bạn không thể dự đoán chúng sẽ hiển thị như thế nào trên các màn hình có kích thước và mật độ khác nhau. Bạn chỉ đang làm điều này để tạm thời di chuyển phần tử, và bạn sẽ điều chỉnh kích thước ở bước tiếp theo.

4. Chọn `button_toast` trong **Component Tree**, nhấp vào tab **Attributes** để mở bảng thuộc tính, sau đó thay đổi `textSize` thành **60sp** (#1 trong hình dưới). Thay đổi `layout_width` thành `wrap_content` (#2 trong hình dưới).

Như hiển thị ở phía bên phải của hình trên (2), bạn có thể nhấp vào bộ điều khiển chiều rộng của trình kiểm tra chế độ xem, xuất hiện dưới dạng hai đoạn ở bên trái và bên phải của ô vuông, cho đến khi nó hiển thị `Wrap Content`. Ngoài ra, bạn cũng có thể chọn **wrap\_content** từ menu `layout_width`.



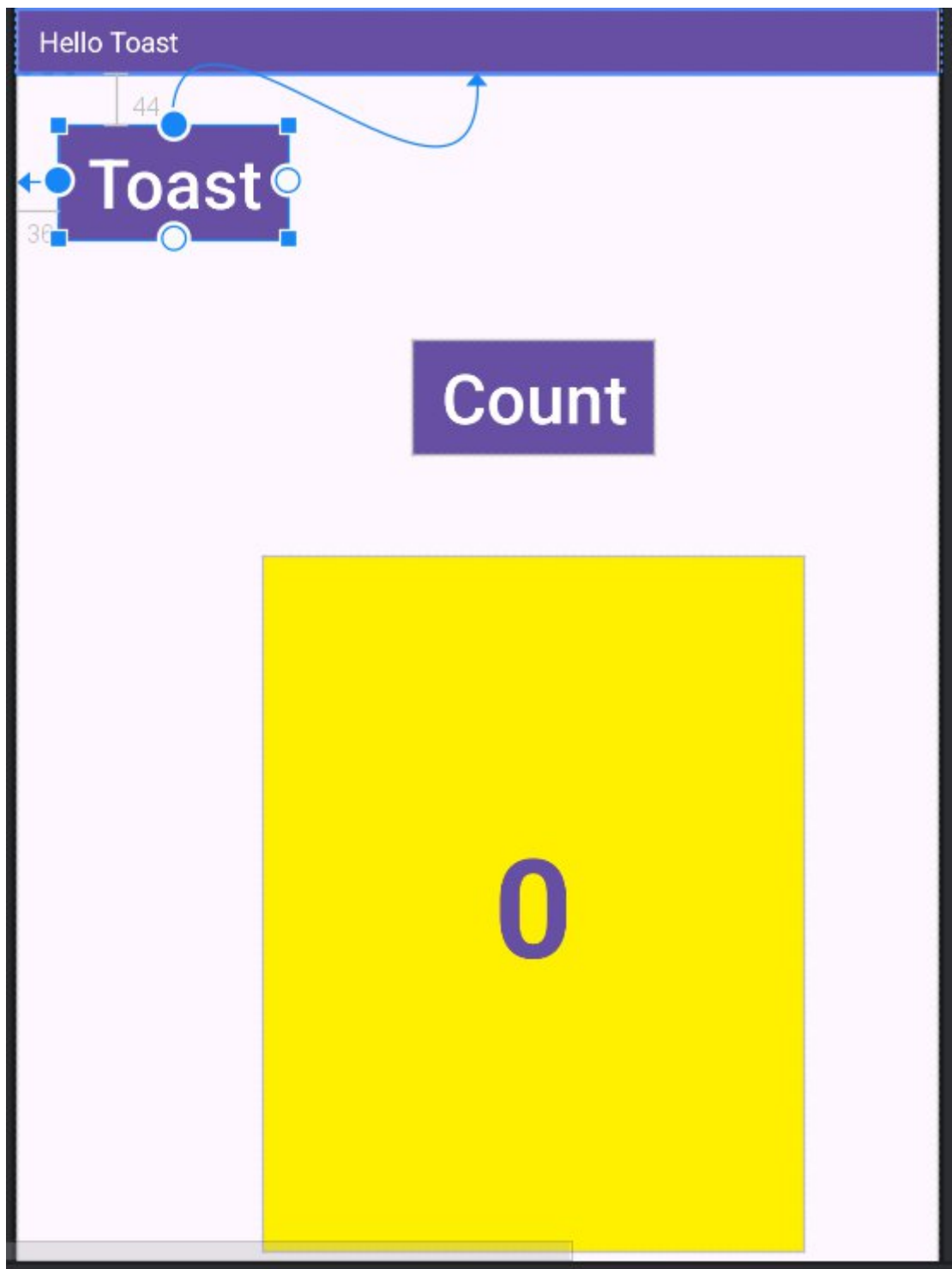
Bạn sử dụng **wrap\_content** để đảm bảo rằng nếu văn bản của nút được dịch sang một ngôn ngữ khác, nút sẽ tự động mở rộng hoặc thu hẹp để phù hợp với từ trong ngôn ngữ đó.

5. Chọn nút `button_count` trong **Component Tree**, thay đổi `textSize` thành **60sp** và `layout_width` thành `wrap_content`, sau đó kéo nút này lên phía trên **TextView** vào một khoảng trống trong bố cục.

## 1.7 Sử dụng ràng buộc đường cơ sở

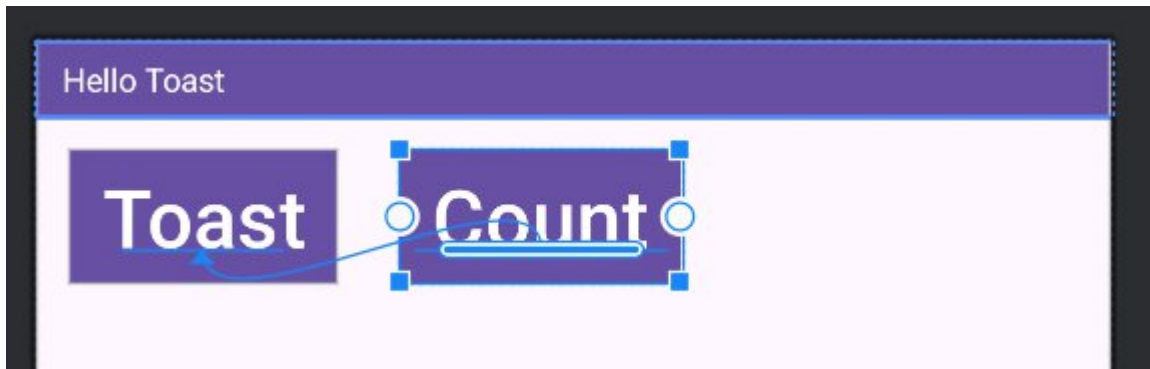
Bạn có thể căn chỉnh một phần tử giao diện người dùng chứa văn bản, chẳng hạn như `TextView` hoặc `Button`, với một phần tử giao diện người dùng khác cũng chứa văn bản. Ràng buộc đường cơ sở cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở của văn bản khớp nhau.

1. Ràng buộc nút `button_toast` vào cạnh trên và cạnh trái của bố cục, kéo nút `button_count` đến một vị trí gần `button_toast`, sau đó ràng buộc `button_count` vào cạnh trái của `button_toast`, như được minh họa trong hình động.



2. Sử dụng ràng buộc đường cơ sở, bạn có thể ràng buộc `button_count` sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của `button_toast`. Chọn phần tử `button_count`, sau đó di chuột qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.

3. Nhấp vào nút ràng buộc đường cơ sở. Tay cầm đường cơ sở sẽ xuất hiện và nhấp nháy màu xanh lục như trong hình động. Nhấp và kéo đường ràng buộc đường cơ sở đến đường cơ sở của phần tử `button_toast`.



## 1.8 Mở rộng các nút theo chiều ngang

Nút gói (pack button) trên thanh công cụ cung cấp các tùy chọn để gom nhóm hoặc mở rộng các phần tử giao diện người dùng được chọn. Bạn có thể sử dụng nó để sắp xếp đều các nút Button theo chiều ngang trên bố cục.

1. Chọn nút `button_count` trong **Component Tree**, sau đó nhấn Shift và chọn nút `button_toast` để chọn cả hai.
2. Nhấp vào nút gói (pack button) trên thanh công cụ và chọn **Expand Horizontally**, như minh họa trong hình dưới đây.
3. Để hoàn thiện bố cục, ràng buộc `show_count` TextView vào cạnh dưới của `button_toast` và vào hai bên cũng như cạnh dưới của bố cục, như trong hình động bên dưới.
4. Bước cuối cùng là thay đổi `layout_width` và `layout_height` của `show_count` **TextView** thành **Match Constraints** và đặt `textSize` thành **200sp**. Bố cục cuối cùng trông giống như hình bên dưới.
5. Nhấp vào nút **Orientation in Editor** trên thanh công cụ và chọn **Switch to Landscape**. Bố cục dành cho máy tính bảng sẽ xuất hiện ở chế độ ngang như hình dưới. (Bạn có thể chọn **Switch to Portrait** để quay lại chế độ dọc.)
6. Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng sau khi chạy ứng dụng để xem giao diện trên các thiết bị khác nhau. Bạn đã tạo thành công một ứng dụng có giao diện phù hợp trên cả điện thoại và máy tính bảng với các kích thước và mật độ màn hình khác nhau.

**1.4) Văn bản và các chế độ cuộn**

**1.5) Tài nguyên có sẵn**

## **Bài 2) Activities**

**2.1) Activity và Intent**

**2.2) Vòng đời của Activity và trạng thái**

**2.3) Intent ngầm định**

## **Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ**

**3.1) Trình gỡ lỗi**

**3.2) Kiểm thử đơn vị**

**3.3) Thư viện hỗ trợ**

## **CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG**

### **Bài 1) Tương tác người dùng**

- 1.1) Hình ảnh có thể chọn
- 1.2) Các điều khiển nhập liệu
- 1.3) Menu và bộ chọn
- 1.4) Điều hướng người dùng
- 1.5) RecyclerView

### **Bài 2) Trải nghiệm người dùng thú vị**

- 2.1) Hình vẽ, định kiểu và chủ đề
- 2.2) Thẻ và màu sắc
- 2.3) Bố cục thích ứng

### **Bài 3) Kiểm thử giao diện người dùng**

- 3.1) Espresso cho việc kiểm tra UI

## **CHƯƠNG 3. LÀM VIỆC TRONG NỀN**

### **Bài 1) Các tác vụ nền**

- 1.1) AsyncTask
- 1.2) AsyncTask và AsyncTaskLoader
- 1.3) Broadcast receivers

### **Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền**

- 2.1) Thông báo
- 2.2) Trình quản lý cảnh báo
- 2.3) JobScheduler

# **CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG**

## **Bài 1) Tùy chọn và cài đặt**

**1.1) Shared preferences**

**1.2) Cài đặt ứng dụng**

## **Bài 2) Lưu trữ dữ liệu với Room**

**2.1) Room, LiveData và ViewModel**

**2.2) Room, LiveData và ViewModel**